Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento Ingeniería Matemática

Optimización No lineal

Tarea 2

Ajuste de modelo SIR para predicciones con datos de Covid

Integrantes: Bruno Hernandez
Benjamín Madariaga
Profesores: Jorge Amaya
Fecha 6 de julio de 2020
Santiago, Chile

1. Resumen

En el contexto de la actual pandemia, y bajo la necesidad de evitar muertes y contagios, se crean esquemas que le permitan a las autoridades poseer parámetros que controlen el proceso, sus efectos sociales y económicos. Una de estas opciones son los modelos matemáticos que permiten crear estimaciones de la cantidad de contagios y muertes que se proyectan a corto plazo.

Los especialistas suponen que en algún momento el número de infectados debe estabilizarse en un máximo desconocido, en un día también desconocido (cuando el incremento se hace nulo o muy cercano a cero). Se modela esto, mediante una función logística de la forma:

$$I(t;\alpha,\beta,\sigma) = \frac{\alpha}{1+\beta e^{-t/\sigma}}.$$

En el siguiente reporte para el curso MA5701-Optimización No Lineal, se ocupará una base de datos con los registros de la cantidad de contagiados y fallecidos en Chile por COVID-19, entre los días 3 de marzo y 1 de junio, del año 2020, para generar un modelo de regresión que explique la curva de crecimiento de estas variables.

Mediante algoritmos de minimización, se buscarán los parámetros que minimicen el error entre nuestro modelo propuesto y los datos reales del registro. Toda esta implimentación se realizará bajo el lenguaje `Python`, más específicamente las librerías `Pandas`, `NumPy` y `SciPy`.

También se estudiará el uso de cotas superiores. Como caso particular, y dado el avance actual de casos, se considerará una cota máxima de 350.000 infectados. Además, considerando una taza promedio de fallecidos por casos confirmados del 7 %, llegamos a una cota de 24.500 defunciones. El uso de estos cotas será estudiado en los puntos 2.c y 2.d.

Se compararán los resultados aplicados a dos métricas para la medición de los errores, el error cuadrático y el error máximo absoluto, y se presentarán las implicancias más importantes que traerán los modelos propuestos como el *peak* de contagios o muertes diarias.

Finalmente, en el último punto, se presentarán los códigos utilizados para la posible recreación del modelo.

2. Modelo

La población total de 18 millones se divide cada día en personas Susceptibles S(t), Infectadas I(t) y Removidas R(t) (estos últimos son los recuperados y fallecidos). Modelaremos a la cantidad de infectados mediante la función logística

$$I(t;\alpha,\beta,\sigma) = \frac{\alpha}{1+\beta e^{-t/\sigma}}$$

Buscaremos estimar los parámetros α, β y σ minimizando el error con los datos observados. Para esto utilizaremos como medidas de error las normas $\|\cdot\|_2$ y $\|\cdot\|_\infty$. Esto es, nuestras funciones objetivo serán mínimos cuadrados y la diferencia máxima.

$$f_2(\alpha,\beta,\sigma) = \sum_{t=1}^{N}\left(I(t) - \frac{\alpha}{1+\beta e^{-t/\sigma}}\right)^2$$
$$f_\infty(\alpha,\beta,\sigma) = \max_{t=[1,\dots,N]}\left|I(t) - \frac{\alpha}{1+\beta e^{-t/\sigma}}\right|$$

Para poder buscar los parámetros óptimos, estudiaremos primero su significado para poder establecer cotas naturales que nos permitan acercarnos a un óptimo global.

En primer lugar vemos que $I(t) \longrightarrow \alpha$ cuando $t \to \infty$, por lo que α sera el máximo de contagiados y, por ende, tendremos que $\alpha > 0$. Dado que $I(t)$ debe ser no negativo (puesto que equivale al conteo de personas) necesitaremos también que $\beta > 0$. Finalmente, notamos que sigma corresponde al periodo de reproducción, por lo que nuevamente sera positivo $\sigma > 0$.

3. Estimación

Dada la naturaleza aleatoria de los datos reales registrados, las funciones de error generadas contienen gran cantidad de oscilaciones, y con esto, gran cantidad de mínimos locales. Esto generó una fuerte dependencia entre el punto inicial utilizado y el resultado devuelto por el algoritmo que suele caer en tales mínimos. Por lo tanto se pierde la certeza de encontrar el mínimo global de la función.

Por esta razón se implementó una función que optimiza el problema para distintos puntos iniciales generados al azar. Se realizaron $N = 10000$ iteraciones, donde en cada iteración se tomaron puntos iniciales según una variable aleatoria χ^2 de parámetros iguales a los valores óptimos encontrados en la iteración anterior. Esto es, si denotamos a_i, b_i, s_i los valores iniciales y $\hat{\alpha}_i, \hat{\beta}_i, \hat{\sigma}_i$ los óptimos encontrados en la i-ésima iteración, tenemos

$$a_i \sim \chi^2(\hat{\alpha}_{i-1}) \qquad , b_i \sim \chi^2(\hat{\beta}_{i-1}) \qquad , s_i \sim \chi^2(\hat{\sigma}_{i-1})$$

La razón de hacer esto cabe en el hecho que en cada iteración el algoritmo encuentra un mínimo local, por lo tanto se partirá nuevamente el algoritmo minimizador desde un punto cercano al anterior (levemente perturbado) con el fin de volver al mismo óptimo, en caso de encontrar un valor óptimo menor, se actualizarán los puntos iniciales y se volverá a perturbar. Esto crea una sucesión minimizante de los parámetros del problema.

a.

Usando la función objetivo f_2 (mínimos cuadrados) se obtuvieron los parámetros óptimos de la siguiente tabla.

Parámetros	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\sigma}$
Valor óptimo	2.433.736,23	4.857,32	16,82

Tabla 3.1: Parámetros óptimos para la función de error f_2.

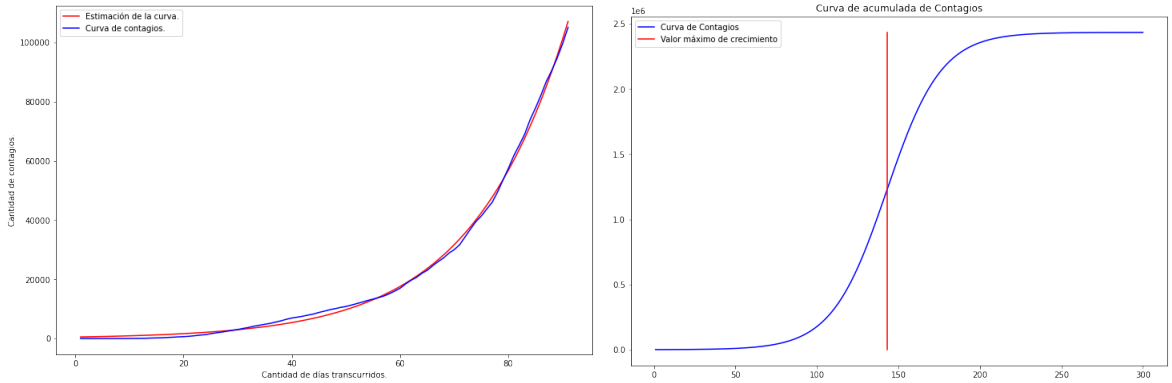


Figura 3.1: Ajuste de los parámetros óptimos a los datos y la proyección indicando el día de más casos nuevos.

Con estos óptimos, se estima que el total de infectados será de 2.433.736. Además, el máximo de infectados nuevos se producirá el 23 de Julio, donde se espera que hayan 36.171 casos nuevos.

b.

Usando la función objetivo f_∞ (diferencia máxima) se obtuvieron los parámetros óptimos de la siguiente tabla.

Parámetros	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\sigma}$
Valor óptimo	430.021,79	1.262,79	15,17

Tabla 3.2: Parámetros óptimos para la función de error f_∞.

Con estos óptimos, se estima que el total de infectados será de 430.022. Además, el máximo de infectados nuevos se producirá el 19 de Junio, donde se espera que hayan 7.086 casos nuevos.

De los resultados de **a.** y **b.** vemos que ambos llegan a valores similares para el periodo de reproducción $\hat{\sigma}$. Sin embargo, las proyecciones en cuanto a cantidad de contagios y peak de casos nuevos diarios difieren ampliamente, siendo estos valores más de 5 veces mayores en los resultados obtenidos usando mínimos cuadrados.

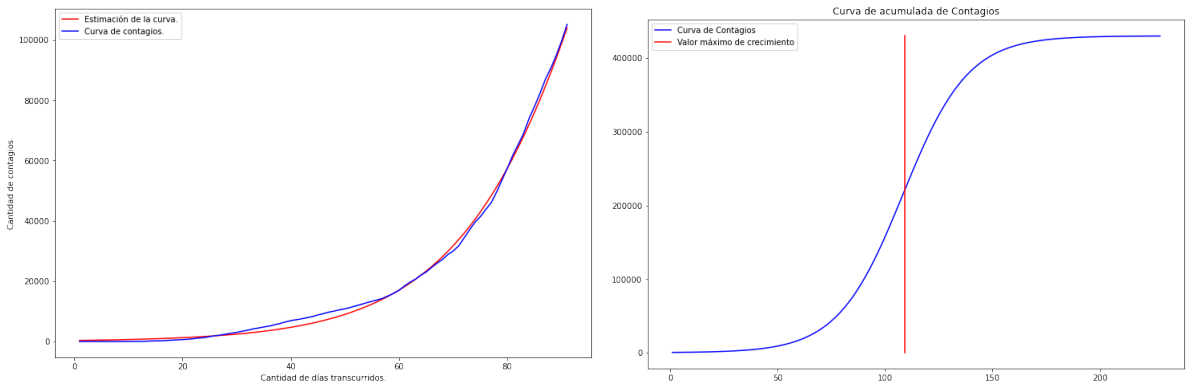


Figura 3.2: Ajuste de los parámetros óptimos a los datos y la proyección indicando el día de más casos nuevos.

c.

Se simuló nuevamente usando las funciones objetivo f_2 y f_∞, pero ahora se acotó superiormente el total de infectados para que no superase $\alpha^{\text{máx}} = 350.000$. A continuación vemos los resultados para ambas funciones de error.

Función de Error	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\sigma}$
f_2	350.000	1.153,87	14,70
f_∞	349.999,99	1.157,41	14.73

Tabla 3.3: Parámetros óptimos encontrados al agregar la restricción $\alpha \leq \alpha^{\text{máx}}$.

Los resultados para ambas funciones fueron muy similares, con diferencias menores al 1 % en todos los parámetros y ajustándose ambas a la cota máxima para casos totales. Esto provoca que sus predicciones para el día de más contagios sean equivalentes, proyectando ambos el día 14 de Junio como el día de más contagios, con valores de 5940 y 5949 contagios para f_2 y f_∞, respectivamente.

d.

Finalmente se repitieron los ejercicios anteriores para estimar la cantidad de defunciones acumuladas. Esto es, ajustaron los parámetros para las funciones de error f_2 y f_∞, cada una con y sin cota superior. Para fijar una cota nos fijamos en la evidencia internacional donde vemos que la tasa de letalidad por contagios difícilmente supera 0,07, por lo que se utilizo la cota $\alpha^{\text{máx}} = 350.000 * 0,07 = 24.500$.

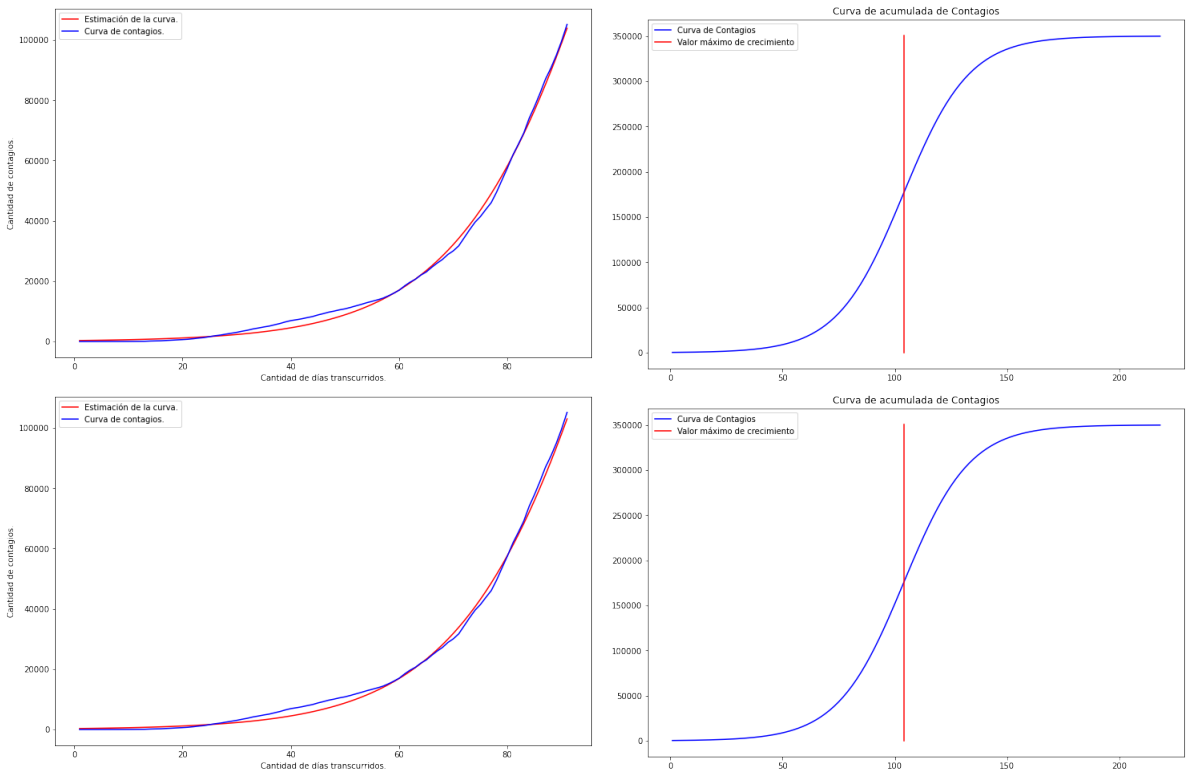


Figura 3.3: Ajuste de los parámetros óptimos a los datos y la proyección indicando el día de más casos nuevos. Arriba usando mínimos cuadrados y abajo usando la diferencia máxima.

Función de Error	$\alpha^{\text{máx}}$	$\hat{\alpha}$	$\hat{\beta}$	$\hat{\sigma}$
f_2	∞	585.680,44	76.952,25	18,29
f_∞	∞	84.505,39	11.098,93	18,34
f_2	24.500	24.500	3.410,44	17.95
f_∞	24.500	20.227,32	2.840,78	17,95

Tabla 3.4: Parámetros óptimos encontrados para la estimación de fallecidos al agregar la restricción $\alpha \leq \alpha^{\text{máx}}$.

Función de Error	$\alpha^{\text{máx}}$	Fecha	Defunciones
f_2	∞	24 de Septiembre	8.002
f_∞	∞	20 de Agosto	1.152
f_2	24.500	26 de Julio	341
f_∞	24.500	23 de Julio	282

Tabla 3.5: Proyección de los días con más defunciones y sus cantidad de fallecidos correspondiente.

Vemos que los resultados obtenidos para el total de defunciones en los casos no acotados son con-

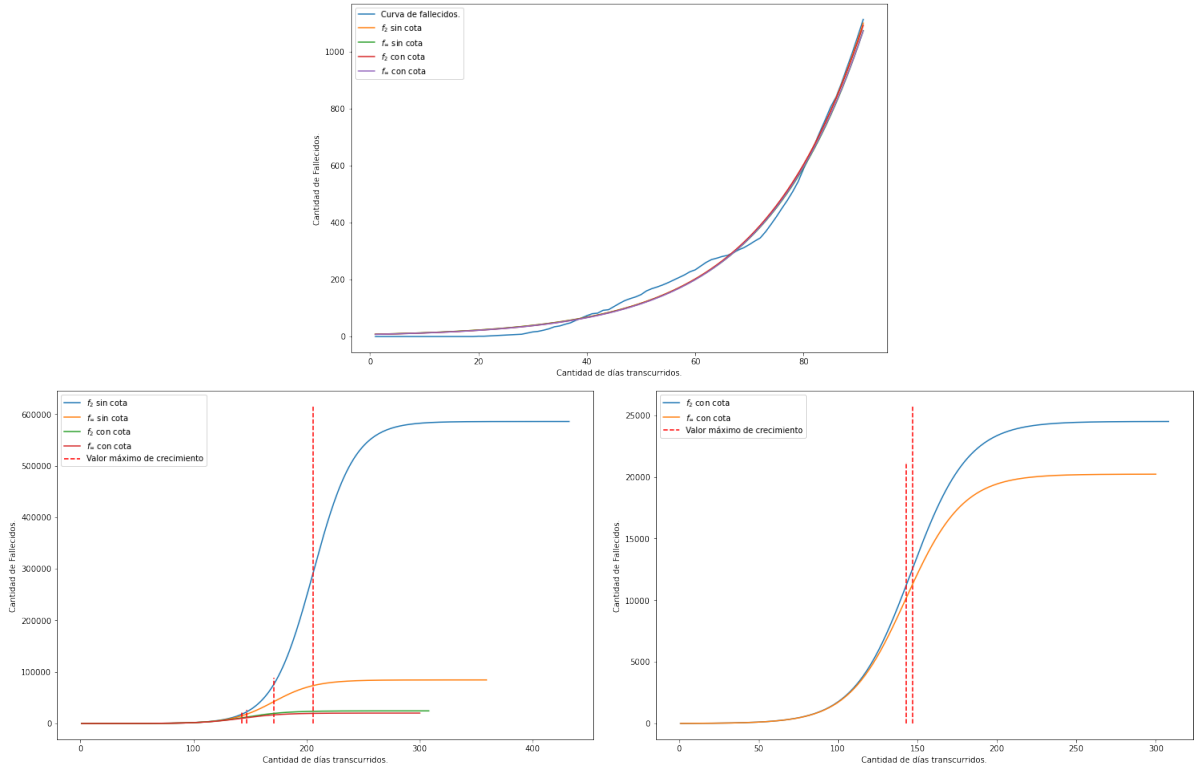


Figura 3.4: Arriba el ajuste de los parámetros óptimos a los datos de fallecidos para todos los métodos, y abajo las proyección indicando el día de más defunciones. A la derecha se hizo un acercamiento a los resultados obtenidos usando cotas superiores.

siderablemente mayores que los obtenidos al usar las cotas superiores. Sin embargo, todos presentan una considerable similitud en la estimación de periodo de reproducción.

4. Conclusiones

A pesar de poseer una función cuyo mínimo es difícil de conseguir algorítmicamente (puesto que las condiciones de optimalidad se consiguen en numerosas ocasiones) es posible conseguir una rutina que se estanque en un valor mínimal. Sin embargo, la certeza de que este valor sea el óptimo global es dependiente del número de iteraciones.

Dado que las métricas para cuantificar el error son diferentes y no existe jerarquía entre estos errores, no podemos decidir qué regresión es mejor, puesto que todas son 'óptimas'. Sin embargo, comparativamente a la situación actual, la regresión que mejor predice el máximo de contagios hasta ahora es aquella que modela con la función de error de diferencia máxima. La veracidad del ajuste conseguido dependerá del la cantidad de datos entregados, por lo que poseemos el margen de escenarios posibles a futuros, el peor escenario con 2.4 millones de contagiados máximos y el mejor escenario con 350 mil.

El algoritmo ocupado aún es perfectible, en el sentido de que los códigos utilizados podría trabajarse para comprometer menor memoria y así, obtener resultados con mayor rapidez.

Por último, en las siguientes páginas se anexan los códigos utilizados.

1 5. Código Python

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from scipy.stats import chi2
```

[2]:
```
df = pd.read_excel('.\datos tarea 2 alumnos.xlsx', header = 1)
df.head()
```

[2]:

	Fecha	Día	Casos totales	Casos recuperados	Fallecidos	Removidos (recup+fallec)
0	2020-03-03	1	1	0	0	0
1	2020-03-04	2	3	0	0	0
2	2020-03-05	3	4	0	0	0
3	2020-03-06	4	5	0	0	0
4	2020-03-07	5	7	0	0	0

[3]:
```
poblacion = 19000000
max_casos = 350000
tasa_mortalidad = 0.07
```

[4]:
```
class MinErrorCOVID:
    """
    Clase optimizadora de la regresión MinErrorCOVID.

    init: MinErrorCOVID(df)
        df corresponde a la base de datos ocupados para el entrenamiento de la␣
 ↪regresión,
        la base debe ser un DataFrame de pandas con, al menos, las siguientes␣
 ↪columnas
        etiquetadas de esta forma:
            -Fecha: columna correspondiente al día en que se tomaron los datos,␣
 ↪en formato
            pandas._libs.tslibs.timestamps.Timestamp

            -Día: valor entero que cuenta los días desde el primer caso␣
 ↪contabilizado (día 1).
```

```
          -Casos totales: valor entero que cuenta los casos totales␣
↪registrados en tal día, considérese este
          como la cantidad total que personas que se encuentran contagiados␣
↪hasta ese día, independientemente
          del día en que se haya registrado el contagio (podría  haberse␣
↪contagiado antes, pero sigue en contagio).

          -Fallecidos: cantidad total de fallecidos registrados hasta esa␣
↪fecha, con causa de muerte asociada a COVID19.


   Generación del modelo en método __modelo__(alpha,beta,sigma, plot, obj):
       alpha, beta, sigma: parámetros.
       plot: True  o False, para mostrar una resumen de los parámetros y una␣
↪gráfica comparativa (defaut = False).
       obj: puede ser 'Casos totales'(default) o 'Fallecidos', entrega el␣
↪objetivo del ajuste.


   Estimación de error en método __error__:
       args:
       - method: funciones de error asociadas.
           1. 'Sqrt': función de error cuadrático.
           2. 'Max': función de  error máximo.


   Descripción del método __ajuste__(method, objetivo, n_iter, lb, ub, dist):
       - itera una cantidad n_iter de veces el proceso de minimización y se␣
↪queda con el modelo de menor error (default = 1000)
       - method es equivalente al del método __error__
       - 'objetivo' es equivalente a 'obj' en el método __modelo__ (default =␣
↪'Casos totales')
       - lb: cotas inferiores de los parámetros, default = [0,0,0]
       - ub: cota superiores de los parámetros, default = [np.inf,np.inf,np.inf]
       - dist: distribución apriori dada a los parámetros (str) (default =␣
↪'Chi2'):
           1. 'Chi2': distribución Chi-square
           2. 'Gamma': distribución Gamma

   Proyección de la fecha de más contagios (requiere que se haya ajustado␣
↪previamente) con método __proyeccion__:
       plot: True  o False, para mostrar una resumen de los parámetros y una␣
↪gráfica comparativa (defaut = False).
       obj: puede ser 'Casos totales'(default) o 'Fallecidos', entrega el␣
↪objetivo del ajuste.
```

```
   Obtención de valores máximos para proyección en método __get_n_max__(),␣
↪requiere que se haya ajustado previamente.
   """

   def __init__(self,df):
       try:
           self._Fecha = df['Fecha']
           self._Dia = df['Día']
           self._Casos = df['Casos totales']
           self._Fallecidos = df['Fallecidos']
           self._ErrorAjuste = None
       except (KeyError,TypeError):
           print('Valor a introducir debe ser pandas.DataFrame correctamente␣
↪etiquetado. ','\nLeer en print(MinErrorCOVID.__doc__)')

   def __modelo__(self,alpha,beta,sigma, plot = False, obj = 'Casos totales',␣
↪fig=False):
       self.alpha = alpha
       self.beta = beta
       self.sigma = sigma
       self.modelo = lambda t: self.alpha/(1 + self.beta*np.exp(-t/self.sigma))
       if plot == True:
           print('Curva de estimación con parámetros:','\n')
           print('Alfa: ',self.alpha,'\n')
           print('Beta: ',self.beta,'\n')
           print('Sigma: ',self.sigma,'\n')
           if fig:
               plt.figure(fig)
           else:
               plt.figure(figsize = (12,8))
           if obj == 'Casos totales':
               plt.plot(self._Dia,self.modelo(self._Dia),'r',label='Estimación␣
↪de la curva.')
               plt.plot(self._Dia,self._Casos, 'b',label='Curva de contagios.')
               plt.xlabel('Cantidad de días transcurridos.')
               plt.ylabel('Cantidad de contagios.')
               plt.legend()
           elif obj == 'Fallecidos':
               plt.plot(self._Dia,self.modelo(self._Dia),'r',label='Estimación␣
↪de la curva.')
               plt.plot(self._Dia,self._Fallecidos, 'b',label='Curva de␣
↪fallecidos.')
               plt.xlabel('Cantidad de días transcurridos.')
               plt.ylabel('Cantidad de Fallecidos.')
               plt.legend()
```

```
    def __error__(self, method, objetivo):
        try:
            if objetivo == 'Casos totales':
                if method == 'Sqrt':
                    self._error = np.sum((self._Casos - self.modelo(self.
↪_Dia))**2)
                    return self._error
                elif method == 'Max':
                    self._error = np.max(np.abs(self._Casos - self.modelo(self.
↪_Dia)))
                    return self._error
                else:
                    print('Método no encontrado.\n')
                    print('ver print(MinErrorCOVID.__doc__) para más información.
↪')
            elif objetivo == 'Fallecidos':
                if method == 'Sqrt':
                    self._error = np.sum((self._Fallecidos - self.modelo(self.
↪_Dia))**2)
                    return self._error
                elif method == 'Max':
                    self._error = np.max(np.abs(self._Fallecidos - self.
↪modelo(self._Dia)))
                    return self._error
                else:
                    print('Método no encontrado.\n')
                    print('ver print(MinErrorCOVID.__doc__) para más información.
↪')
        except:
            print('Incertar un modelo primero.')

    def __ajuste__(self, method, objetivo = 'Casos totales', n_iter = 1000, lb =␣
↪[0,0,0], ub = [poblacion,np.inf,np.inf], dist = 'Chi2'):
        cotas = list(zip(lb,ub))
        F = lambda v,t: v[0]/(1 + v[1]*np.exp(-t/v[2]))
        if objetivo == 'Casos totales':
            if method == 'Sqrt':
                f2 = lambda v: np.sum((self._Casos - F(v,self._Dia))**2)
            elif method == 'Max':
                f2 = lambda v: np.max(np.abs(self._Casos - F(v,self._Dia)))
            else:
                print('Método inválido')
        elif objetivo == 'Fallecidos':
            if method == 'Sqrt':
                f2 = lambda v: np.sum((self._Fallecidos - F(v,self._Dia))**2)
            elif method == 'Max':
```

```
                f2 = lambda v: np.max(np.abs(self._Fallecidos - F(v,self._Dia)))
            else:
                print('Método inválido')
        else:
            print('Objetivo inválido')


        try:
            a,b,c = 10,10,10
            for i in range(n_iter):
                self.__modelo__(a,b,c, obj = objetivo)
                error1 = self.__error__(method, objetivo)
                alf = self.__generate_random__(self.alpha, dist)
                bet = self.__generate_random__(self.beta, dist)
                sig = self.__generate_random__(self.sigma, dist)
                m = minimize(f2,[alf,bet,sig], options={'maxiter':1000}, bounds␣
↪= cotas)
                alf2,bet2,sig2 = m.x
                self.__modelo__(alf2,bet2,sig2,obj = objetivo)
                error2 = self.__error__(method, objetivo)
                if error2 < error1:
                    a,b,c  = alf2,bet2,sig2
            print('Parámetros establecidos para modelo de predicción de␣
↪',objetivo,'\n')
            print('Error mínimo encontrado: ',self._error,'\n')
            self.__modelo__(a,b,c,  plot=True, obj = objetivo)
        except:
            print('Proceso detenido.')

    def __generate_random__(self, parameter, dist):
        if dist == 'Chi2':
            if parameter!=0:
                param = chi2.rvs(parameter, size=1)[0]
            else:
                param = chi2.rvs(10, size=1)[0]
        elif dist == 'Gamma':
            if parameter!=0:
                param = np.random.gamma(parameter, size=1)[0]
            else:
                param = np.random.gamma(10, size=1)[0]
        return param

    def __proyeccion__(self, plot = False, objetivo = 'Casos totales'):
        I_prim = lambda t: (self.alpha*self.beta*np.exp(-t/self.sigma))/(self.
↪sigma*(1 +self.beta*np.exp(-t/self.sigma))**2)

        self.__get_n_max__()
```

```
        D2 = np.arange(1,int(self.n_max*2.1)+1)

        L = np.linspace(0,np.max(I_prim(D2))*1.05,len(D2),dtype=int)
        N = self.n_max*np.ones(len(D2))

        if objetivo == 'Casos totales':
            data_label = 'Contagios'
        elif objetivo == 'Fallecidos':
            data_label = 'Fallecidos'
        else:
            raise ValueError('Objeto {} no definido dentro de los valores␣
↪esperados'.format(objetivo))


        print('El día de más {} es: {}\nY la cantidad de {} es de: {}\n'.
↪format(data_label,self.n_max,data_label,self.dif_max))
        dia_critico = df['Fecha'][0] + np.timedelta64(self.n_max-1,'D')
        print('El día de más {} será el {}\ndel mes {}'.
↪format(data_label,dia_critico.day,dia_critico.month))

        if plot:
            plt.figure(1)
            plt.figure(figsize = (12,8))
            plt.plot(D2,I_prim(D2),color = 'b', label = '{}'.format(data_label))
            plt.plot(N,L, color = 'r', label = 'Valor máximo')
            plt.title('Curva de Crecimiento de {}'.format(data_label))
            plt.legend()

            L2 = np.linspace(0,np.max(self.modelo(D2))*1.05,len(D2),dtype=int)
            plt.figure(2)
            plt.figure(figsize = (12,8))
            plt.plot(D2,self.modelo(D2),color = 'b', label = 'Curva de {}'.
↪format(data_label))
            plt.plot(N,L2, color = 'r', label = 'Valor máximo de crecimiento')
            plt.title('Curva de acumulada de {}'.format(data_label))
            plt.legend()

    def __get_n_max__(self):
        self.dif_max = 0
        self.n_max = 0
        for i in range(1,1000):
            if self.modelo(i)-self.modelo(i-1)> self.dif_max:
                self.n_max = i
                self.dif_max = self.modelo(i)-self.modelo(i-1)
```

[5]:
```
print(MinErrorCOVID.__doc__)
```

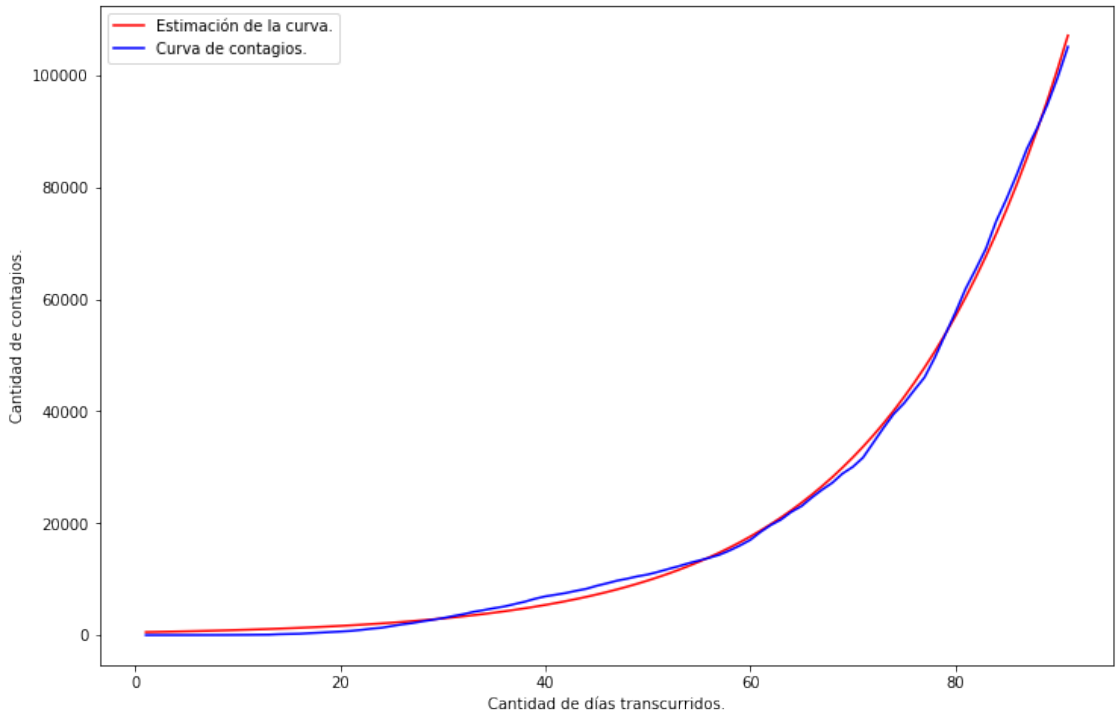
```
    Clase optimizadora de la regresión MinErrorCOVID.

    init: MinErrorCOVID(df)
        df corresponde a la base de datos ocupados para el entrenamiento de la
regresión,
        la base debe ser un DataFrame de pandas con, al menos, las siguientes
columnas
        etiquetadas de esta forma:
            -Fecha: columna correspondiente al día en que se tomaron los datos,
en formato
            pandas._libs.tslibs.timestamps.Timestamp

            -Día: valor entero que cuenta los días desde el primer caso
contabilizado (día 1).

            -Casos totales: valor entero que cuenta los casos totales
registrados en tal día, considérese este
            como la cantidad total que personas que se encuentran contagiados
hasta ese día, independientemente
            del día en que se haya registrado el contagio (podría  haberse
contagiado antes, pero sigue en contagio).

            -Fallecidos: cantidad total de fallecidos registrados hasta esa
fecha, con causa de muerte asociada a COVID19.


    Generación del modelo en método __modelo__(alpha,beta,sigma, plot, obj):
        alpha, beta, sigma: parámetros.
        plot: True  o False, para mostrar una resumen de los parámetros y una
gráfica comparativa (defaut = False).
        obj: puede ser 'Casos totales'(default) o 'Fallecidos', entrega el
objetivo del ajuste.


    Estimación de error en método __error__:
        args:
        - method: funciones de error asociadas.
            1. 'Sqrt': función de error cuadrático.
            2. 'Max': función de  error máximo.


    Descripción del método __ajuste__(method, objetivo, n_iter, lb, ub, dist):
        - itera una cantidad n_iter de veces el proceso de minimización y se
queda con el modelo de menor error (default = 1000)
        - method es equivalente al del método __error__
        - 'objetivo' es equivalente a 'obj' en el método __modelo__ (default =
'Casos totales')
```

```
        - lb: cotas inferiores de los parámetros, default = [0,0,0]
        - ub: cota superiores de los parámetros, default =
[np.inf,np.inf,np.inf]
        - dist: distribución apriori dada a los parámetros (str) (default =
'Chi2'):
            1. 'Chi2': distribución Chi-square
            2. 'Gamma': distribución Gamma

    Proyección de la fecha de más contagios (requiere que se haya ajustado
previamente) con método __proyeccion__:
        plot: True  o False, para mostrar una resumen de los parámetros y una
gráfica comparativa (defaut = False).
        obj: puede ser 'Casos totales'(default) o 'Fallecidos', entrega el
objetivo del ajuste.

    Obtención de valores máximos para proyección en método __get_n_max__(),
requiere que se haya ajustado previamente.
```

[6]:
```
Error = MinErrorCOVID(df)
```

[7]:
```
Error.__ajuste__('Sqrt',n_iter=10000)
```

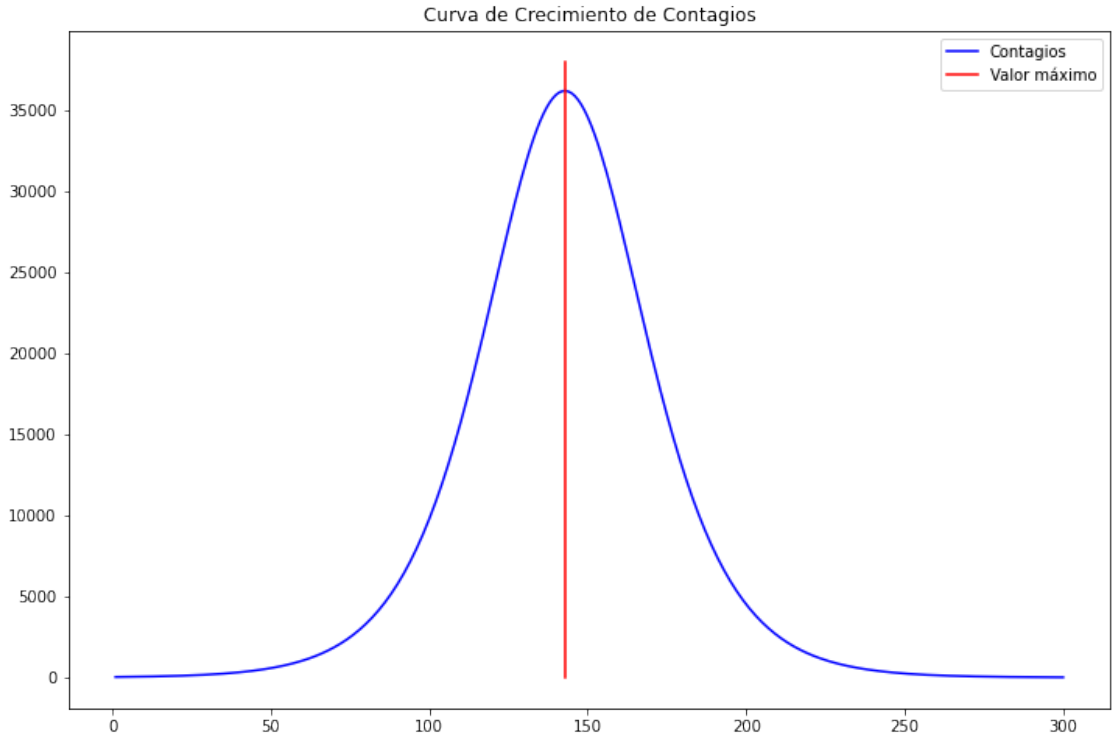
```
Parámetros establecidos para modelo de predicción de  Casos totales

Error mínimo encontrado:  101173282.02024227

Curva de estimación con parámetros:

Alfa:  2433686.355214544

Beta:  4857.233270961953

Sigma:  16.818745040553257
```

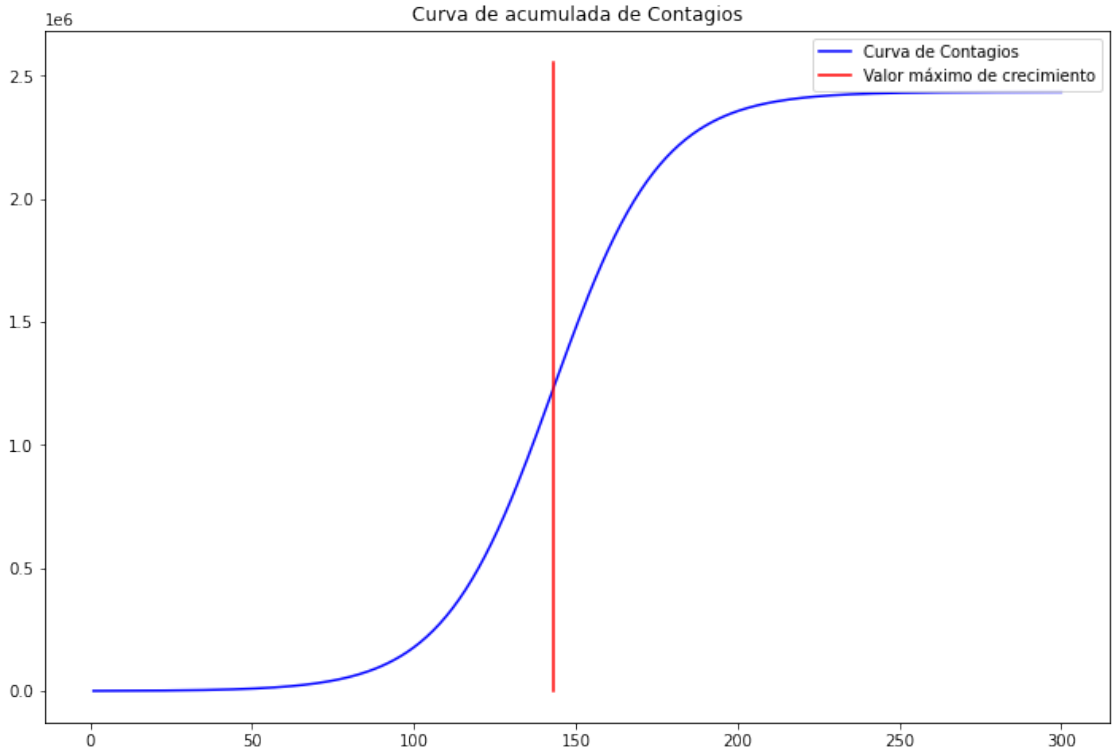


[8]:
```
Error.__proyeccion__(plot=True)
```

```
El día de más Contagios es: 143
Y la cantidad de Contagios es de: 36170.36138768005

El día de más Contagios será el 23
del mes 7
```

```
<Figure size 432x288 with 0 Axes>
```

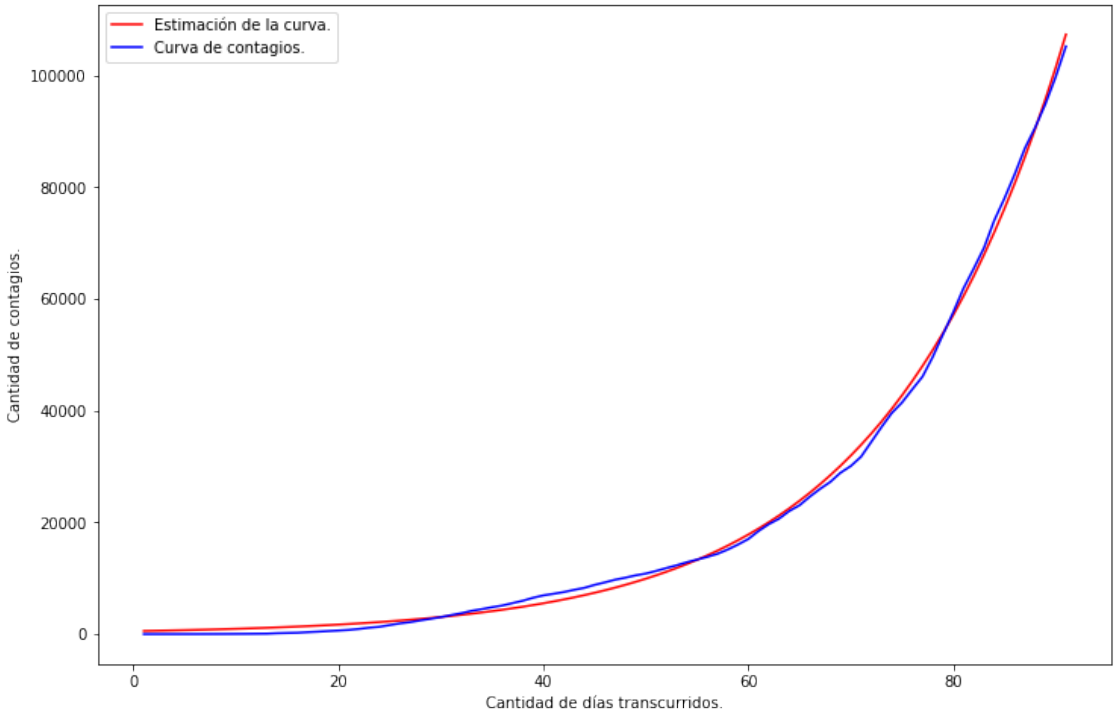

Curva de Crecimiento de Contagios
Contagios
Valor máximo
35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
300


Curva de acumulada de Contagios
1e6
Curva de Contagios
Valor máximo de crecimiento
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300

[9]:
```
Error.__ajuste__('Max',n_iter=10000)
```

```
C:\Users\paula\Anaconda3\lib\site-packages\pandas\core\series.py:679:
RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

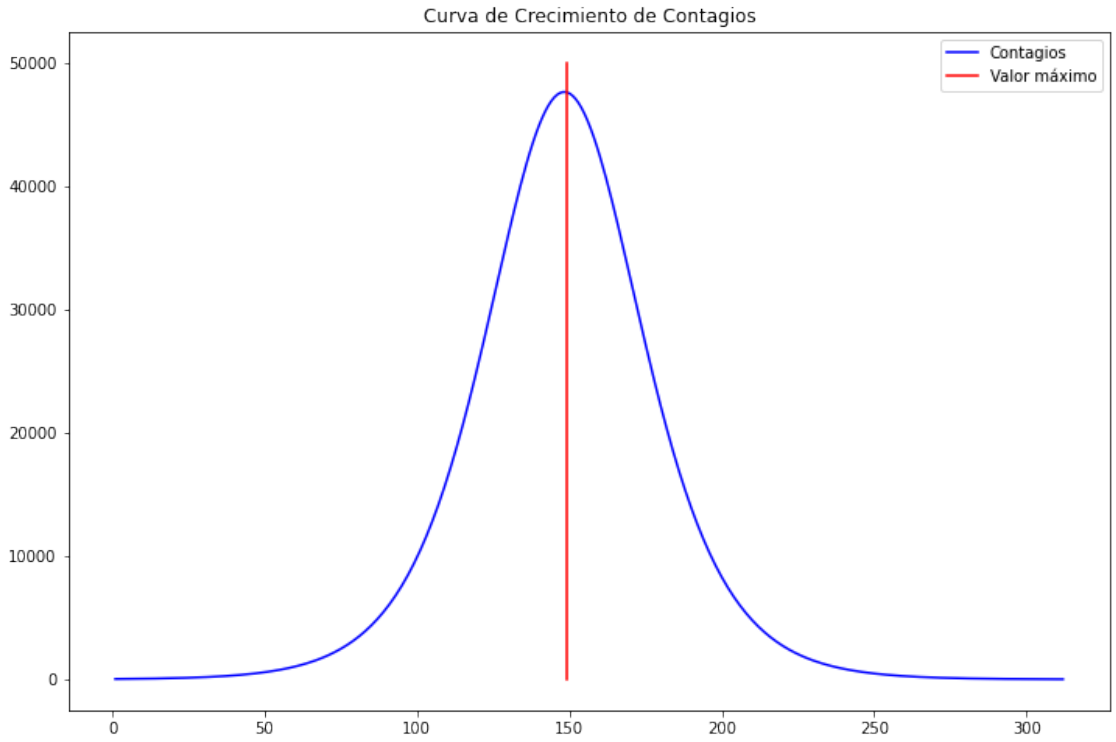
```
Parámetros establecidos para modelo de predicción de  Casos totales

Error mínimo encontrado:  2150.332632933867

Curva de estimación con parámetros:

Alfa:  3237826.6902359347

Beta:  6173.090324316205

Sigma:  16.995082583058853
```

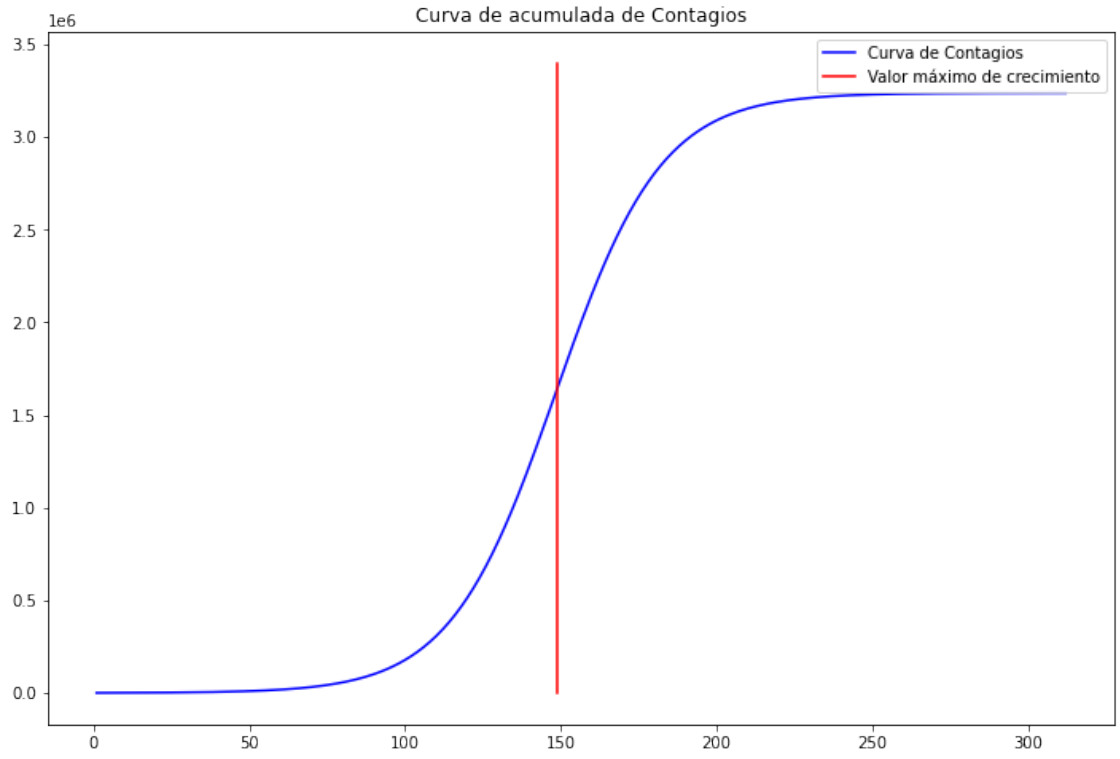


[10]:
```
Error.__proyeccion__(plot=True)
```

```
El día de más Contagios es: 149
Y la cantidad de Contagios es de: 47624.28151116031

El día de más Contagios será el 29
```

```
del mes 7
<Figure size 432x288 with 0 Axes>
```





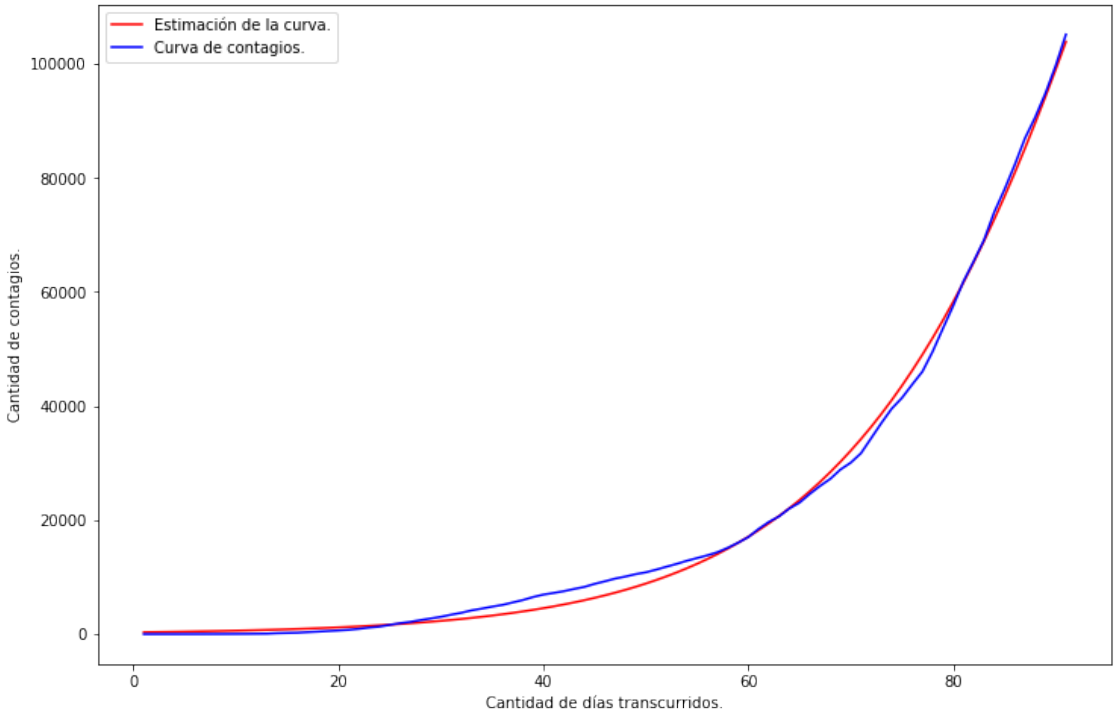
[11]:
```
Error.__ajuste__('Sqrt', n_iter=10000, ub = [max_casos,np.inf,np.inf])
Error.__proyeccion__(plot=True)
Error.__ajuste__('Max', n_iter=10000, ub = [max_casos,np.inf,np.inf])
Error.__proyeccion__(plot=True)
```

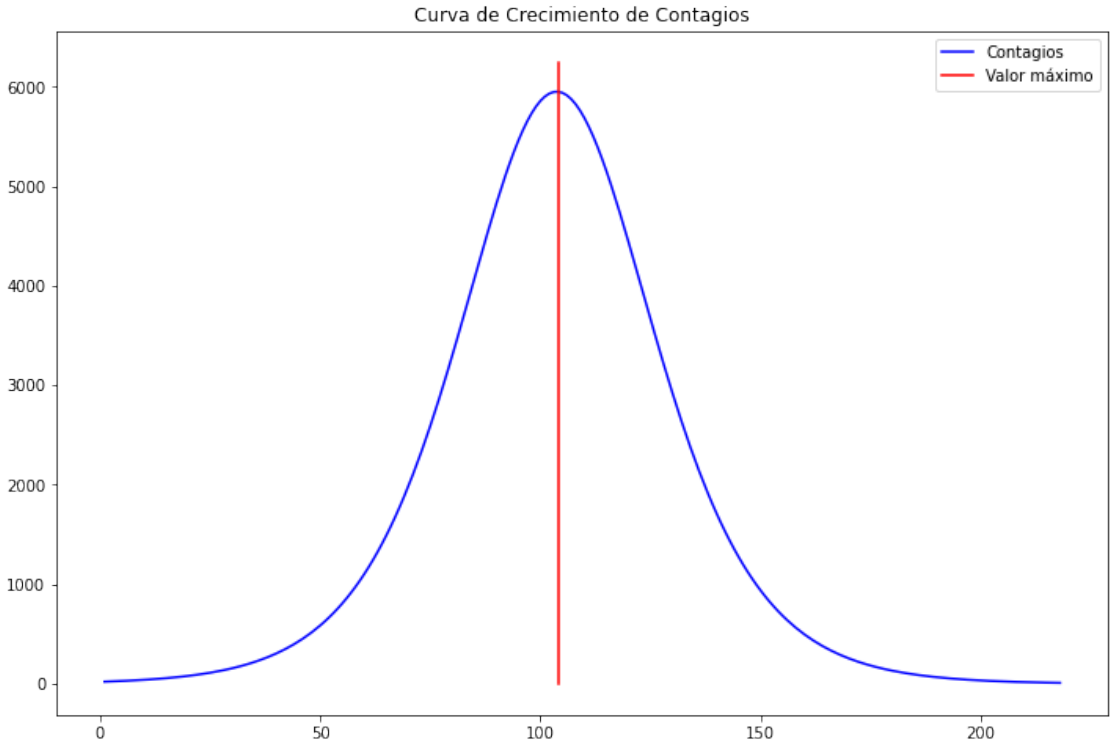
```
C:\Users\paula\Anaconda3\lib\site-packages\pandas\core\series.py:679:
RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)
Parámetros establecidos para modelo de predicción de  Casos totales

Error mínimo encontrado:  166424666.29275757

Curva de estimación con parámetros:

Alfa:  350000.0

Beta:  1153.8707260936378

Sigma:  14.704635072024757

El día de más Contagios es: 104
Y la cantidad de Contagios es de: 5949.707174131967
```

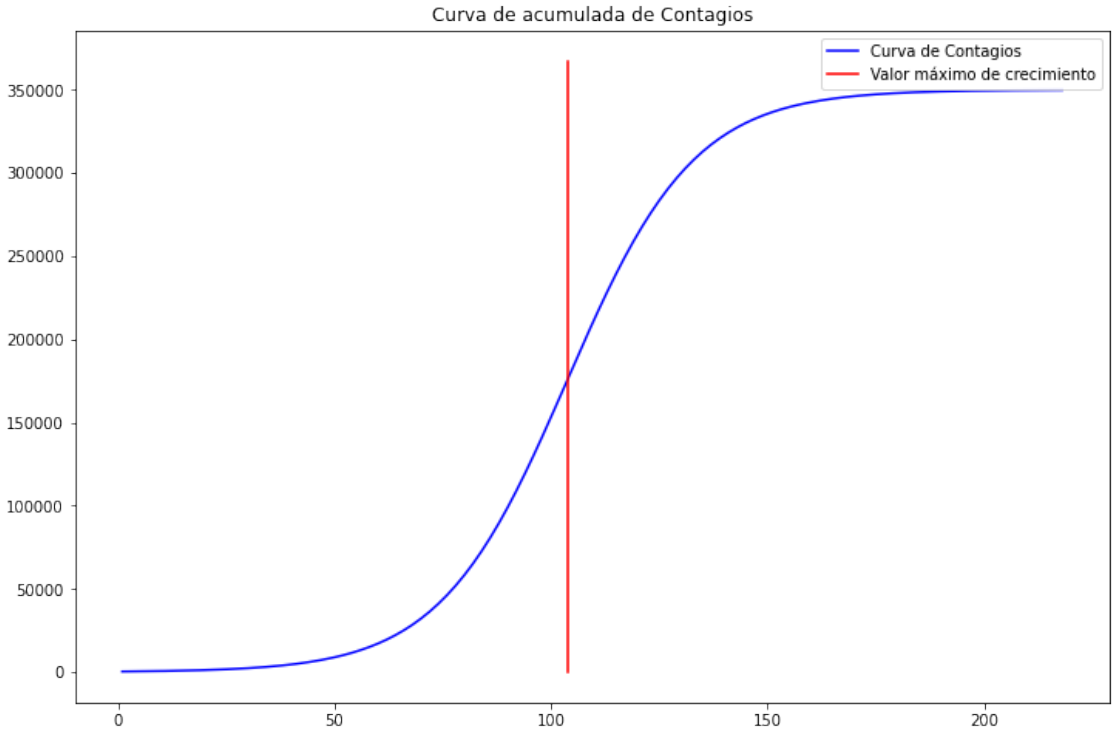
```
El día de más Contagios será el 14
del mes 6
```

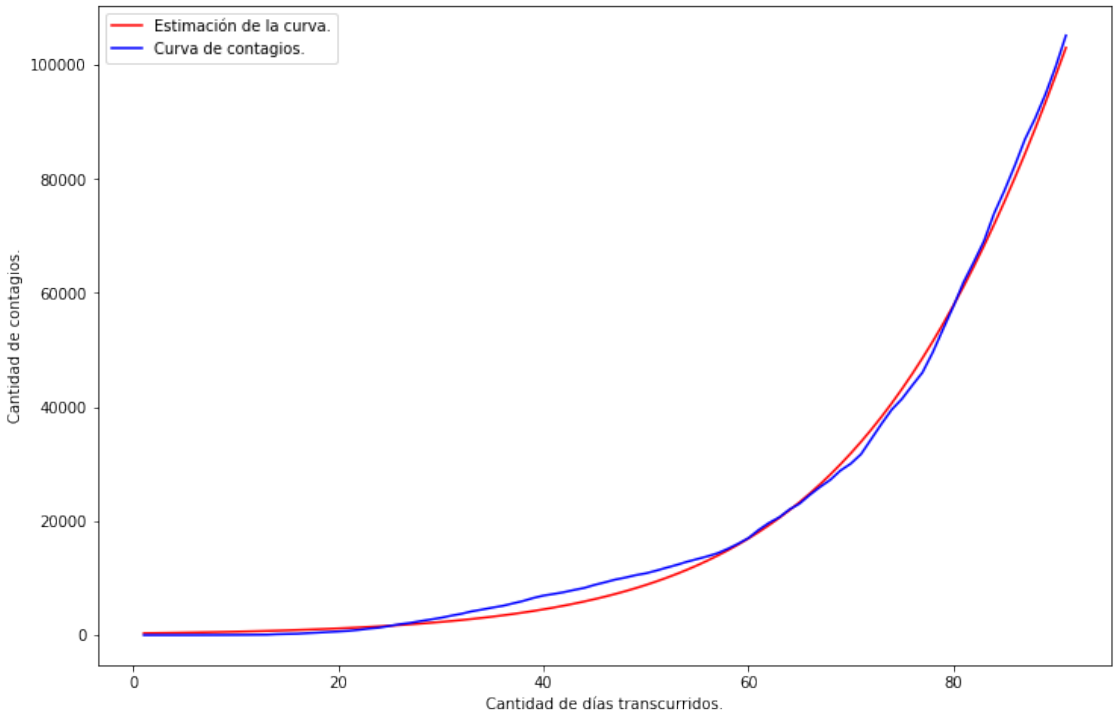
```
C:\Users\paula\Anaconda3\lib\site-packages\pandas\core\series.py:679:
RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

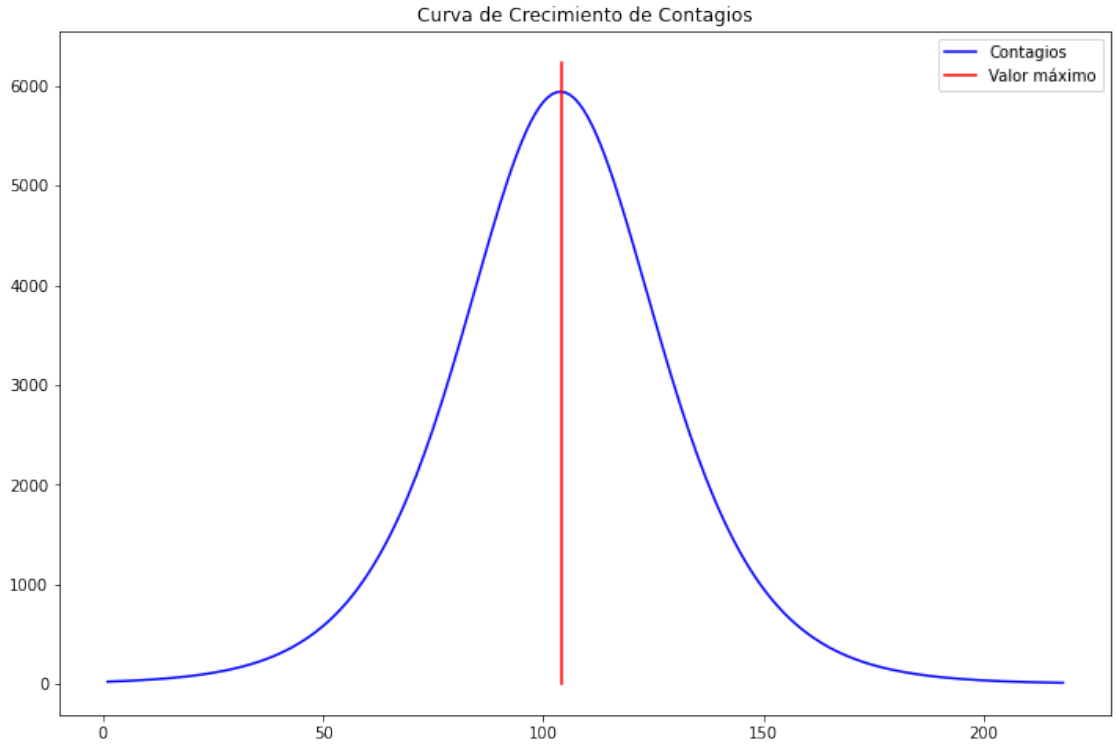
```
Parámetros establecidos para modelo de predicción de  Casos totales

Error mínimo encontrado:  2564.0774776097896

Curva de estimación con parámetros:

Alfa:  349999.839205091

Beta:  1157.376915411714

Sigma:  14.726371398692828

El día de más Contagios es: 104
Y la cantidad de Contagios es de: 5940.167311973404

El día de más Contagios será el 14
del mes 6
```

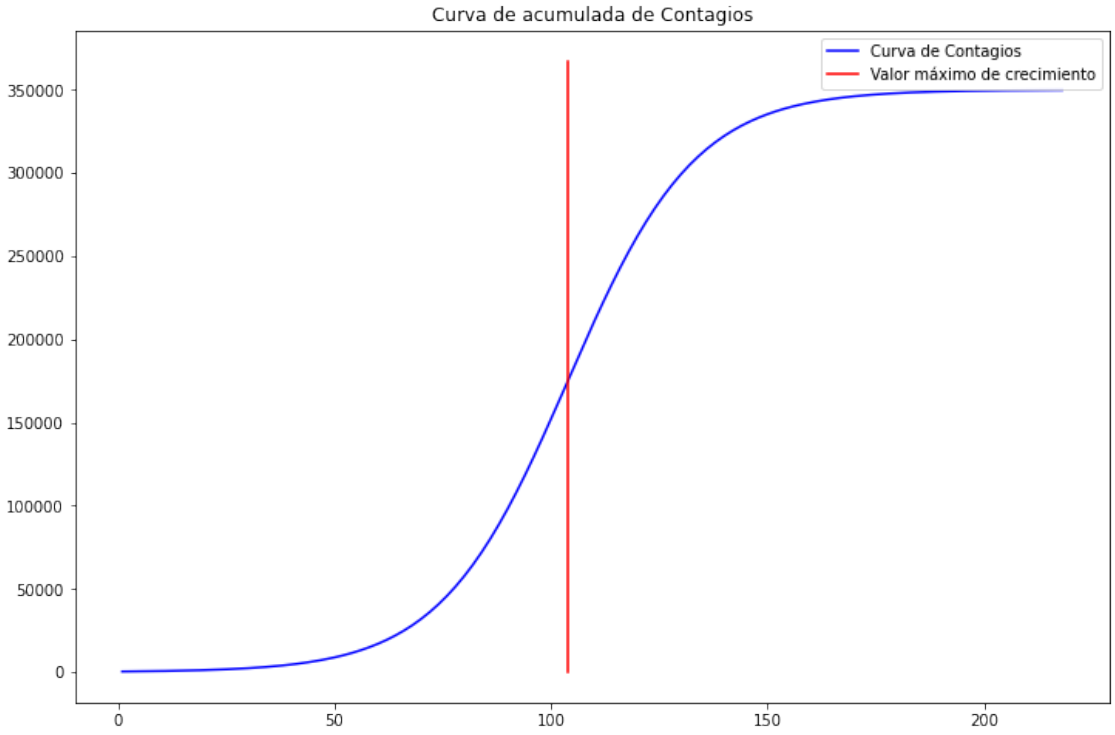



Curva de Crecimiento de Contagios
Contagios
Valor máximo
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200


Curva de acumulada de Contagios
Curva de Contagios
Valor máximo de crecimiento
350000
300000
250000
200000
150000
100000
50000
0
0
50
100
150
200


Estimación de la curva.
Curva de contagios.
Cantidad de contagios.
Cantidad de días transcurridos.
0
20000
40000
60000
80000
100000
0
20
40
60
80

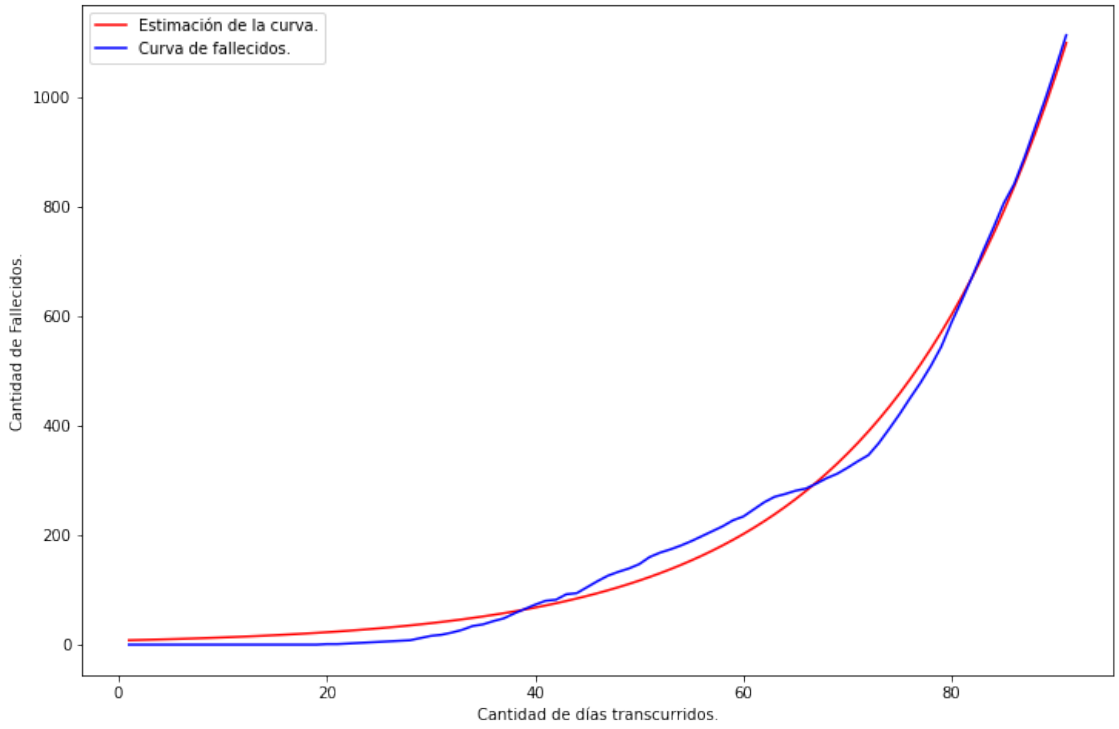

Curva de Crecimiento de Contagios
Contagios
Valor máximo
0
1000
2000
3000
4000
5000
6000
0
50
100
150
200

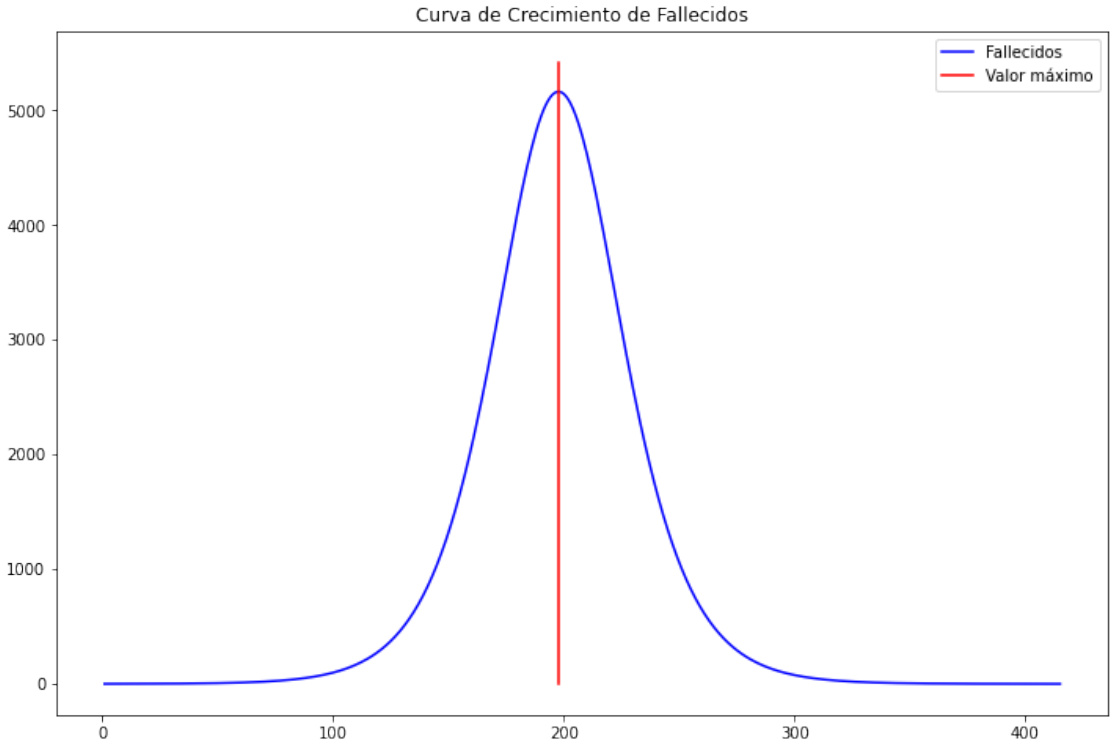


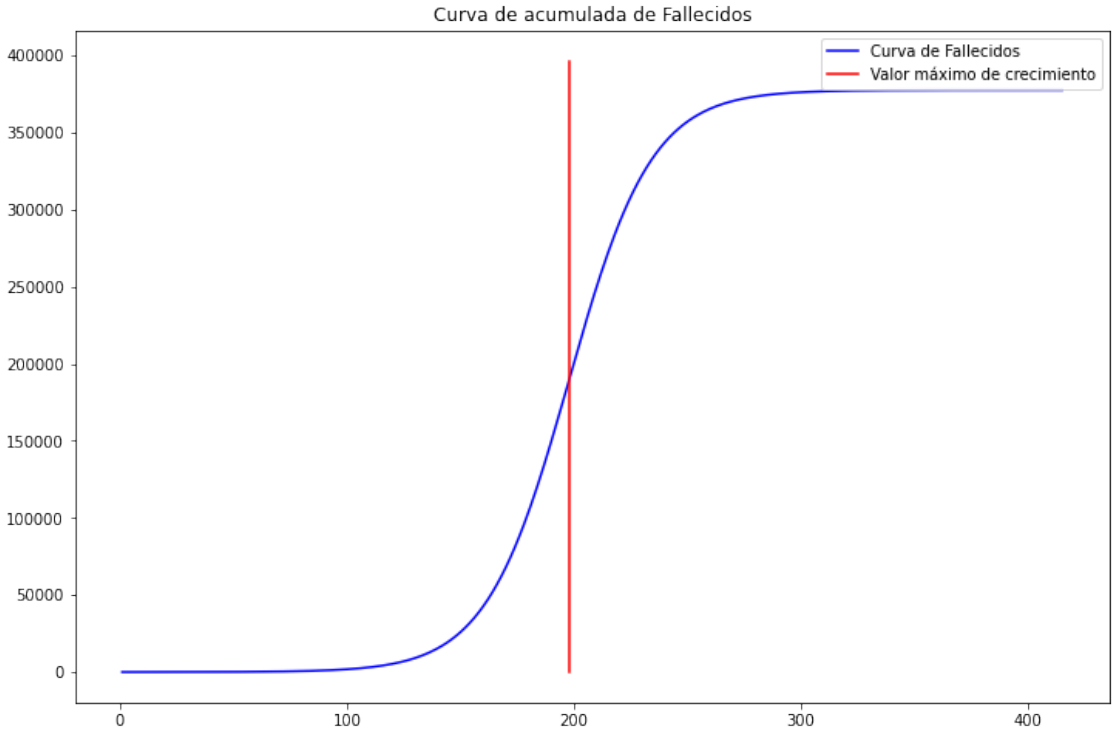
[12]:
```
Error.__ajuste__('Sqrt', objetivo = 'Fallecidos', n_iter=5000)
Error.__proyeccion__(plot=True, objetivo = 'Fallecidos')
Error.__ajuste__('Max', objetivo = 'Fallecidos', n_iter=5000)
Error.__proyeccion__(plot=True, objetivo = 'Fallecidos')
```

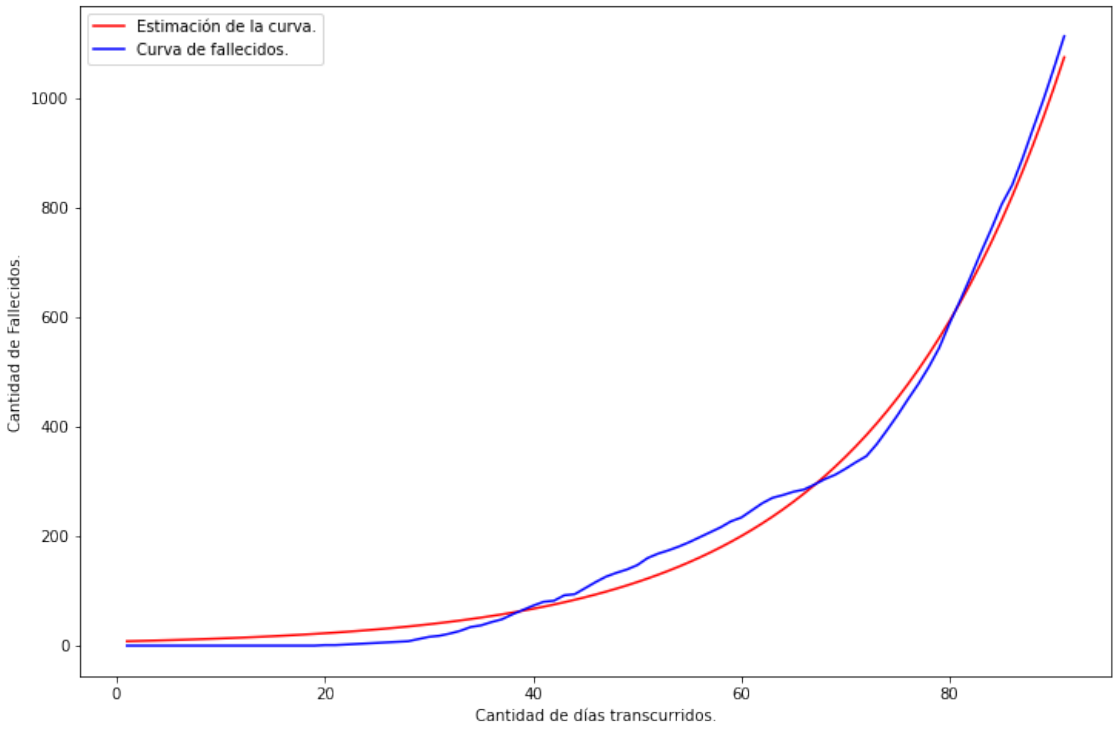
```
Parámetros establecidos para modelo de predicción de  Fallecidos

Error mínimo encontrado:  47949.79881229669

Curva de estimación con parámetros:

Alfa:  377537.6642759233

Beta:  49732.06472884228

Sigma:  18.281287809394964

El día de más Fallecidos es: 198
Y la cantidad de Fallecidos es de: 5162.419353612408

El día de más Fallecidos será el 16
del mes 9

C:\Users\paula\Anaconda3\lib\site-packages\pandas\core\series.py:679:
```

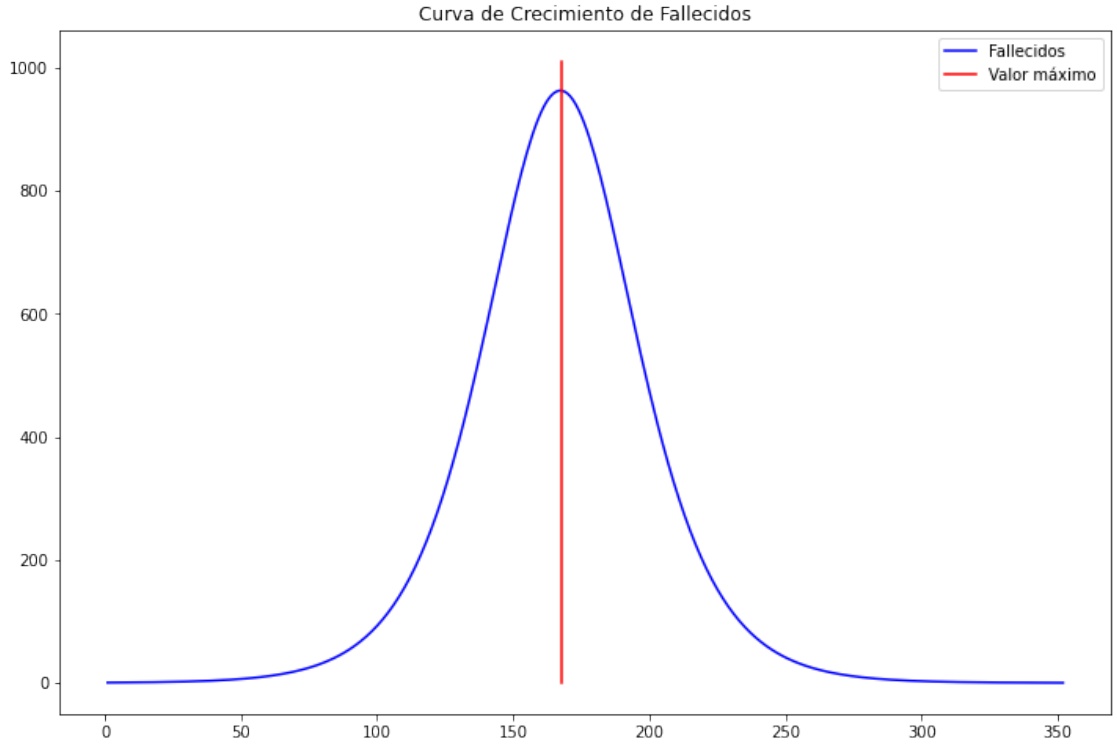
```
RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)
Parámetros establecidos para modelo de predicción de  Fallecidos

Error mínimo encontrado:  38.55328821100363

Curva de estimación con parámetros:

Alfa:  70557.66264231205

Beta:  9303.917657510914

Sigma:  18.31376014681857

El día de más Fallecidos es: 168
Y la cantidad de Fallecidos es de: 963.1032702804368

El día de más Fallecidos será el 17
del mes 8
```

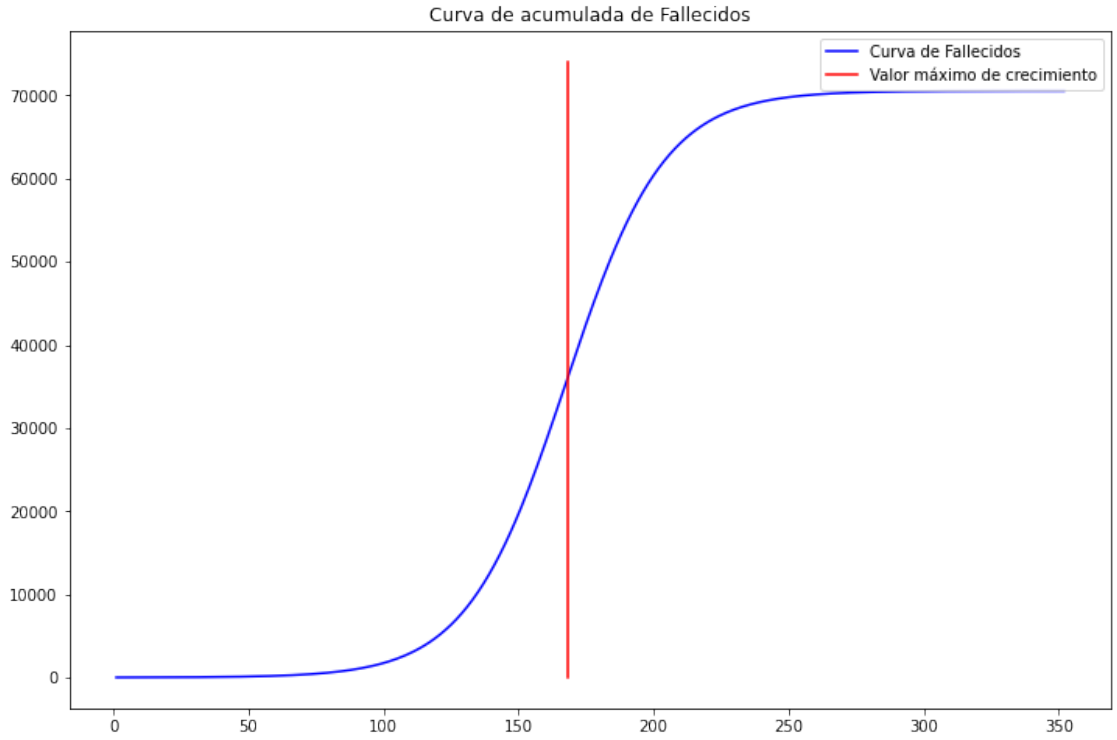



Curva de Crecimiento de Fallecidos
Fallecidos
Valor máximo
5000
4000
3000
2000
1000
0
0
100
200
300
400


Curva de acumulada de Fallecidos
Curva de Fallecidos
Valor máximo de crecimiento
400000
350000
300000
250000
200000
150000
100000
50000
0
0
100
200
300
400


Estimación de la curva.
Curva de fallecidos.
Cantidad de Fallecidos.
Cantidad de días transcurridos.
0
200
400
600
800
1000
0
20
40
60
80

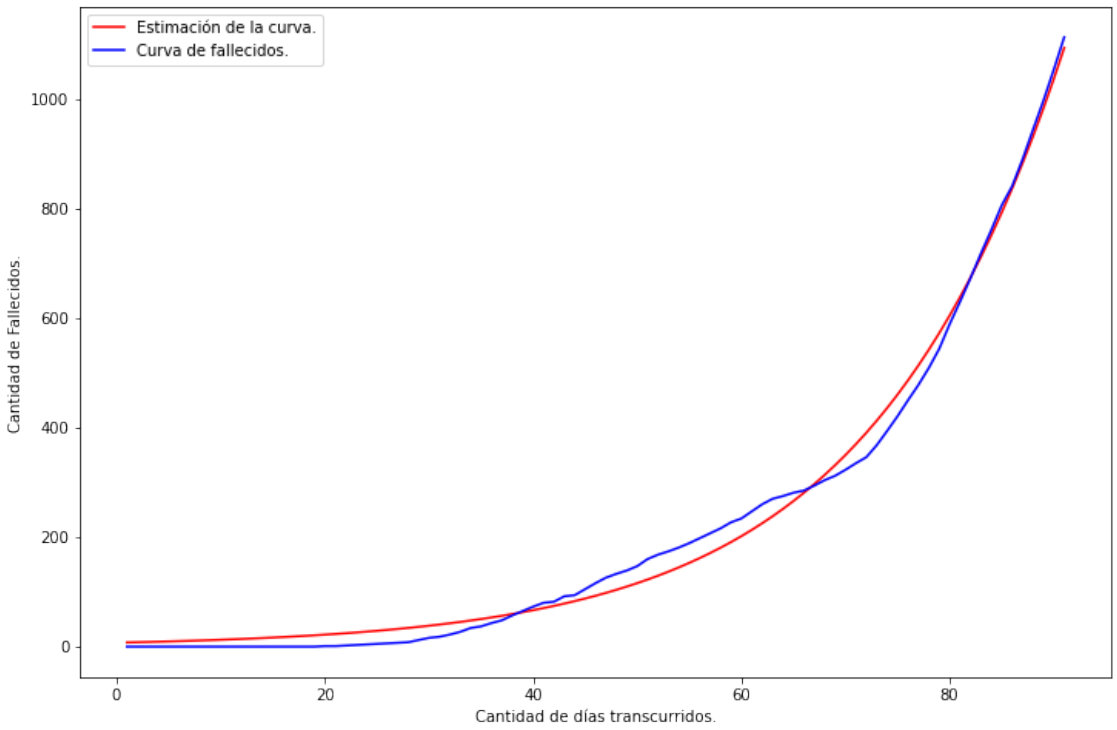

Curva de Crecimiento de Fallecidos
Fallecidos
Valor máximo
0
200
400
600
800
1000
0
50
100
150
200
250
300
350

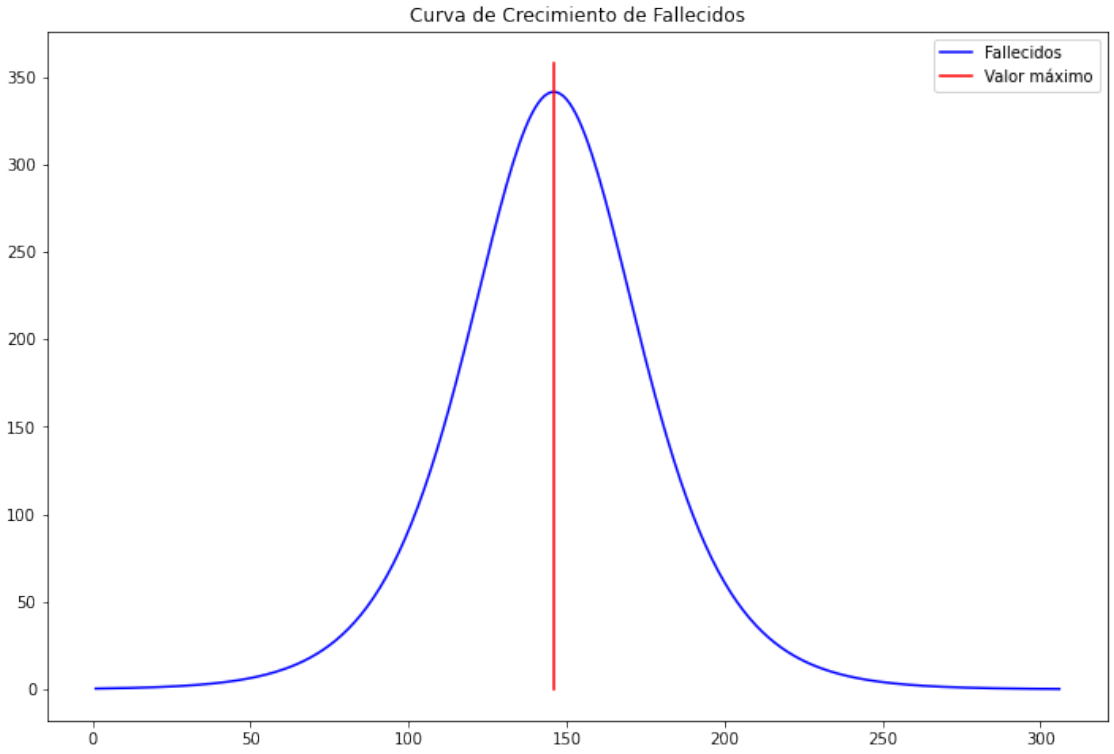


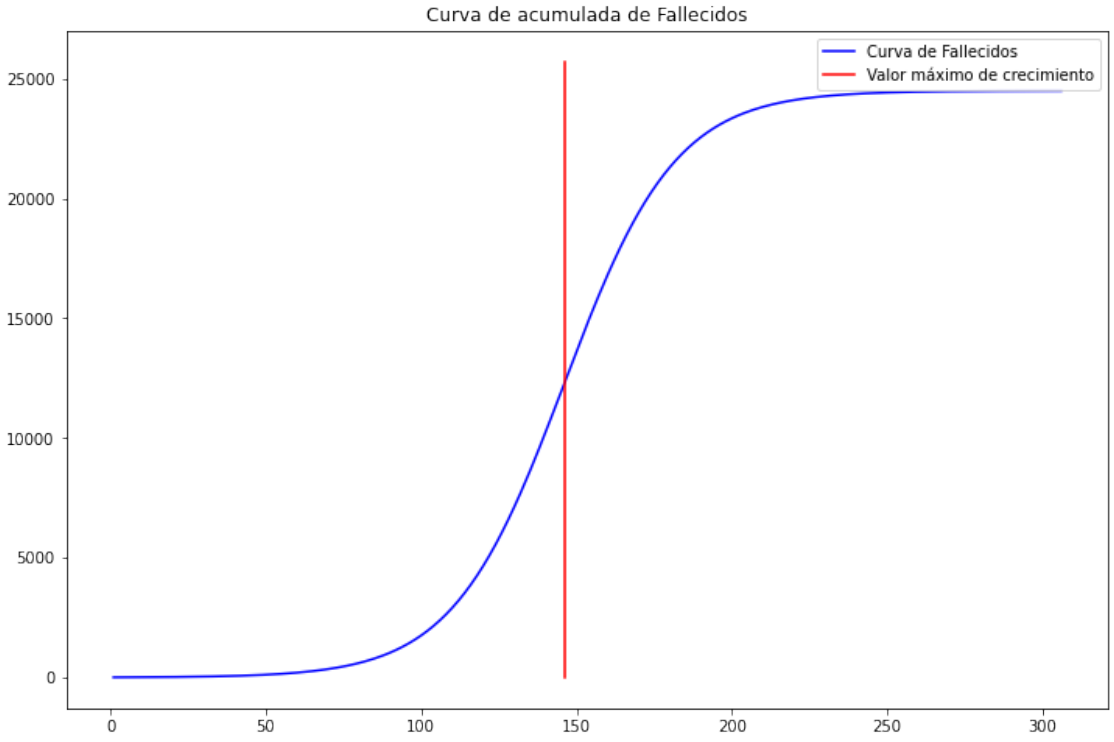
[13]:
```
Error.__ajuste__('Sqrt', objetivo = 'Fallecidos', n_iter=5000, ub =␣
 ↪[max_casos*tasa_mortalidad,np.inf,np.inf])
Error.__proyeccion__(plot=True, objetivo = 'Fallecidos')
Error.__ajuste__('Max', objetivo = 'Fallecidos', n_iter=5000, ub =␣
 ↪[max_casos*tasa_mortalidad,np.inf,np.inf])
Error.__proyeccion__(plot=True, objetivo = 'Fallecidos')
```

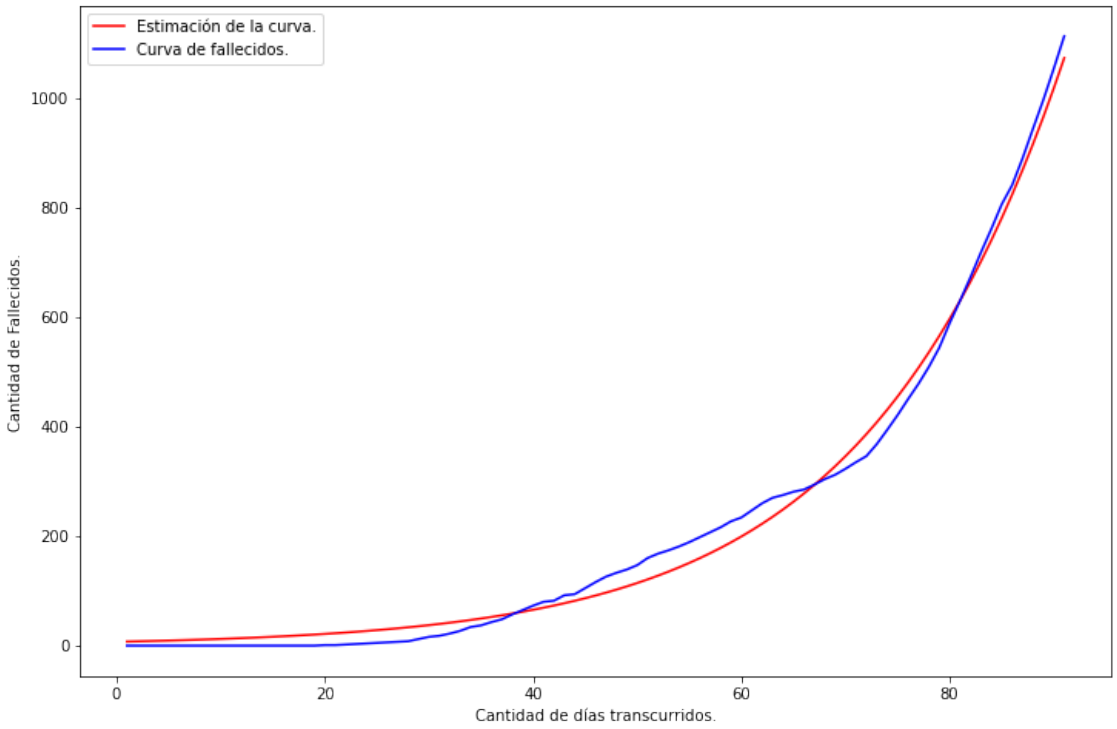
```
Parámetros establecidos para modelo de predicción de  Fallecidos

Error mínimo encontrado:  49463.848844014945

Curva de estimación con parámetros:

Alfa:  24500.000000000004

Beta:  3410.4397160228573

Sigma:  17.94547036367185

El día de más Fallecidos es: 146
Y la cantidad de Fallecidos es de: 341.2288668615656

El día de más Fallecidos será el 26
```

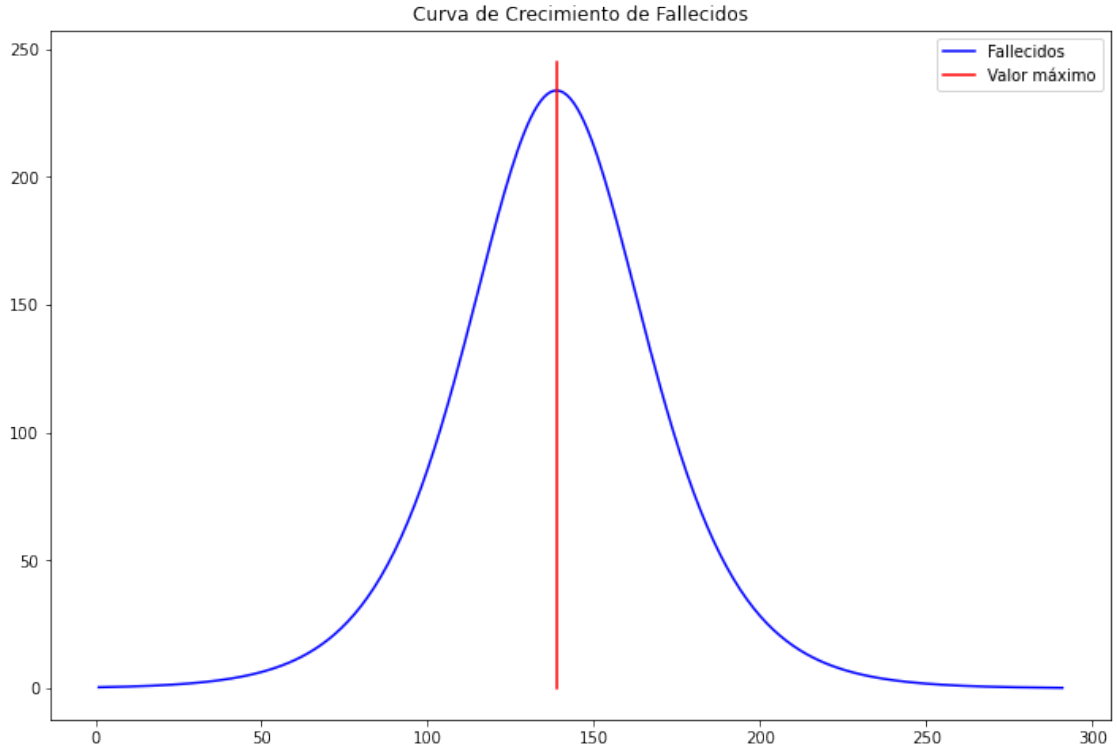
```
del mes 7

C:\Users\paula\Anaconda3\lib\site-packages\pandas\core\series.py:679:
RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)

Parámetros establecidos para modelo de predicción de  Fallecidos

Error mínimo encontrado:  40.16231037432874

Curva de estimación con parámetros:

Alfa:  16677.96650715506

Beta:  2388.2526031666284

Sigma:  17.838836074023384

El día de más Fallecidos es: 139
Y la cantidad de Fallecidos es de: 233.70380484019006

El día de más Fallecidos será el 19
del mes 7
```

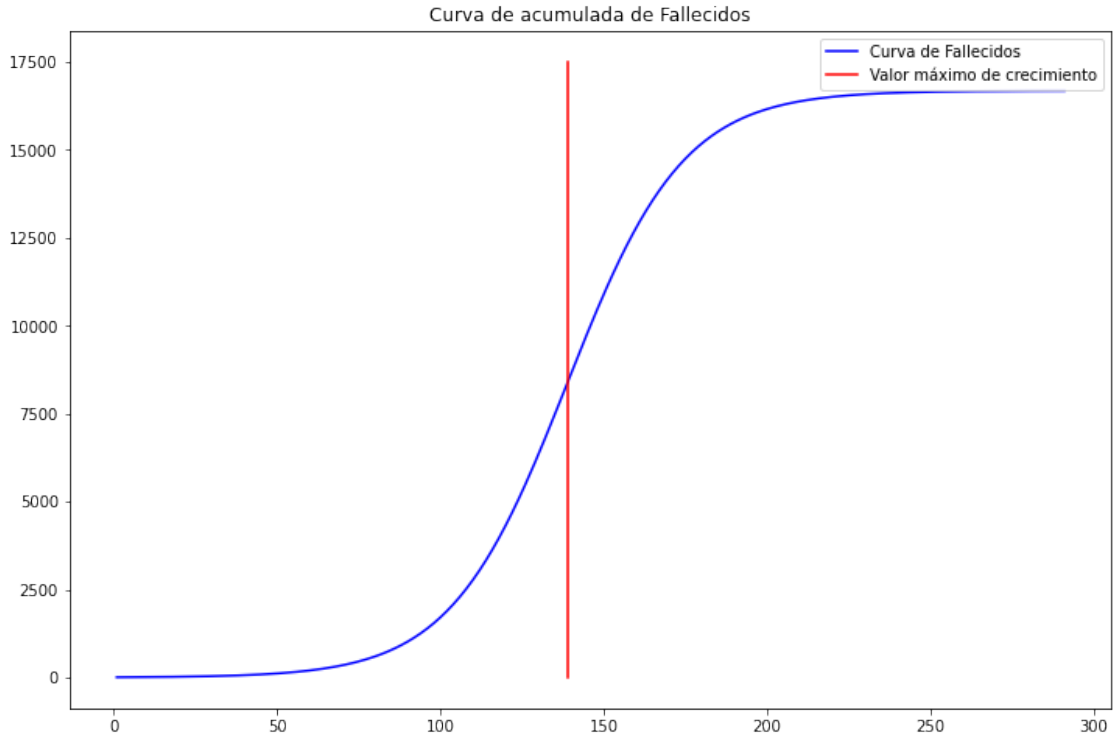



Curva de Crecimiento de Fallecidos
Fallecidos
Valor máximo
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300


Curva de acumulada de Fallecidos
Curva de Fallecidos
Valor máximo de crecimiento
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
300

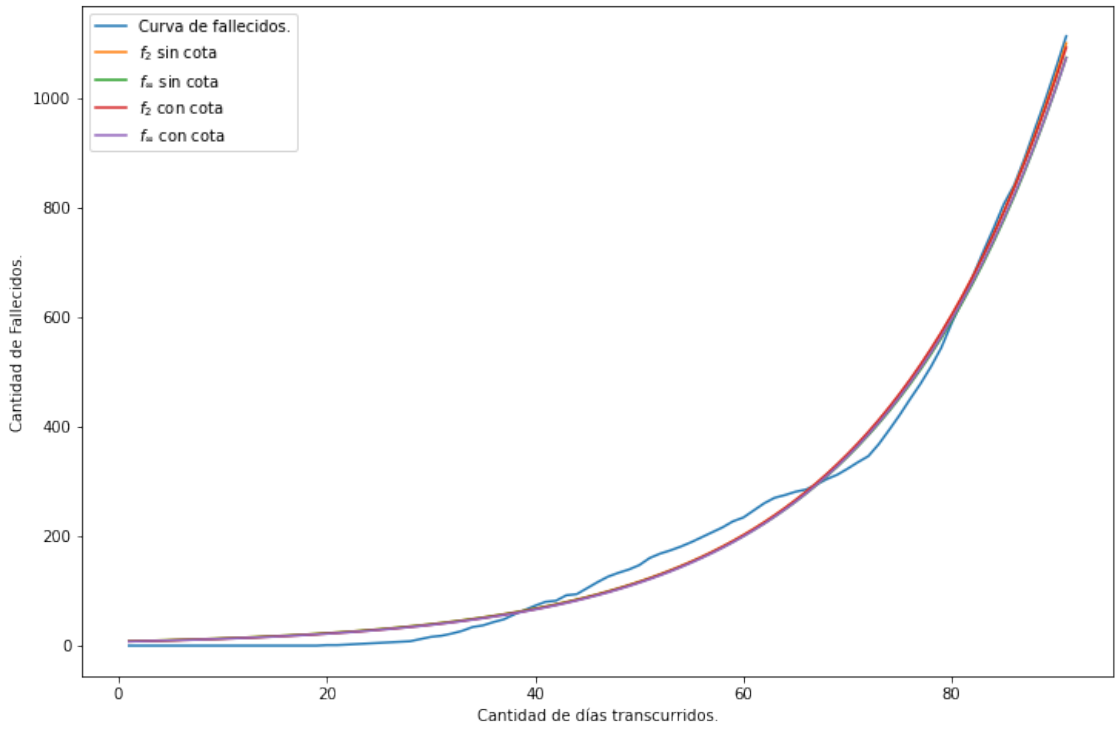

Estimación de la curva.
Curva de fallecidos.
Cantidad de Fallecidos.
Cantidad de días transcurridos.
0
200
400
600
800
1000
0
20
40
60
80


Curva de Crecimiento de Fallecidos
Fallecidos
Valor máximo
0
50
100
150
200
250
0
50
100
150
200
250
300



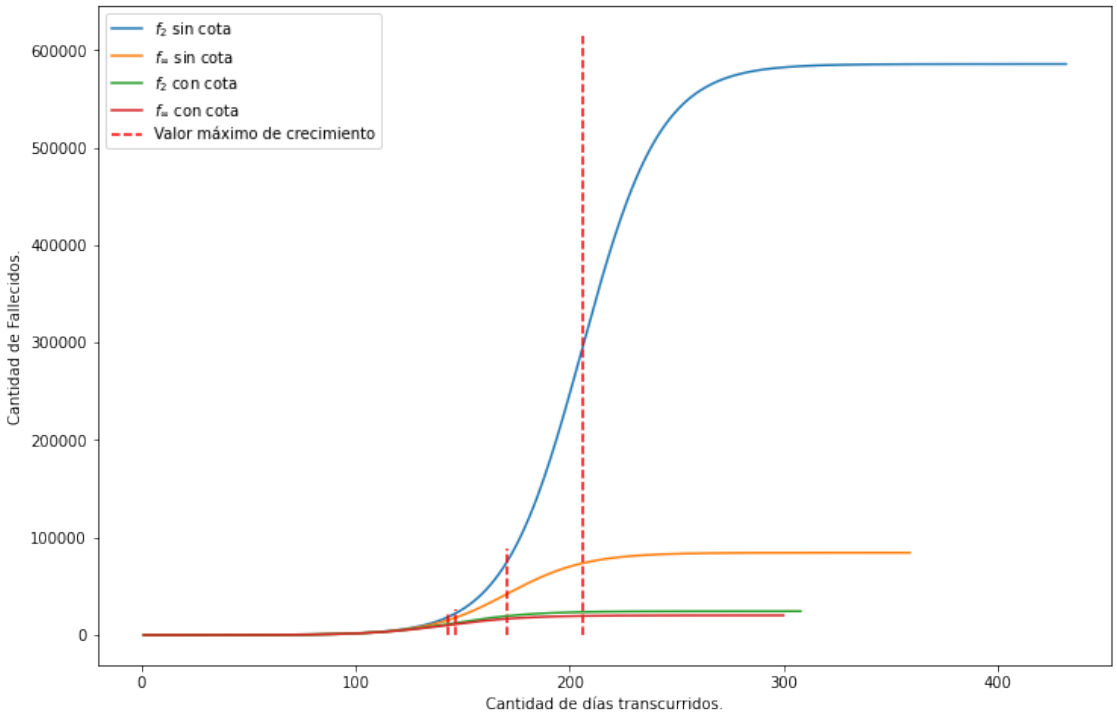
[14]:
```
plt.figure(figsize = (12,8))
plt.plot(Error._Dia,Error._Fallecidos,label='Curva de fallecidos.')

Error.__modelo__(585680.44 , 76952.25, 18.29, obj='Fallecidos')
plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_2$ sin cota')
Error.__modelo__(84505.39, 11098.93, 18.34, obj='Fallecidos')
plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_\infty$ sin cota')
Error.__modelo__(24500, 3410.44, 17.95, obj='Fallecidos')
plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_2$ con cota')
Error.__modelo__(20227.32, 2840.78, 17.95, obj='Fallecidos')
plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_\infty$ con cota')


plt.xlabel('Cantidad de días transcurridos.')
plt.ylabel('Cantidad de Fallecidos.')
plt.legend()
```

[14]:
```
<matplotlib.legend.Legend at 0x1b3f75243c8>
```



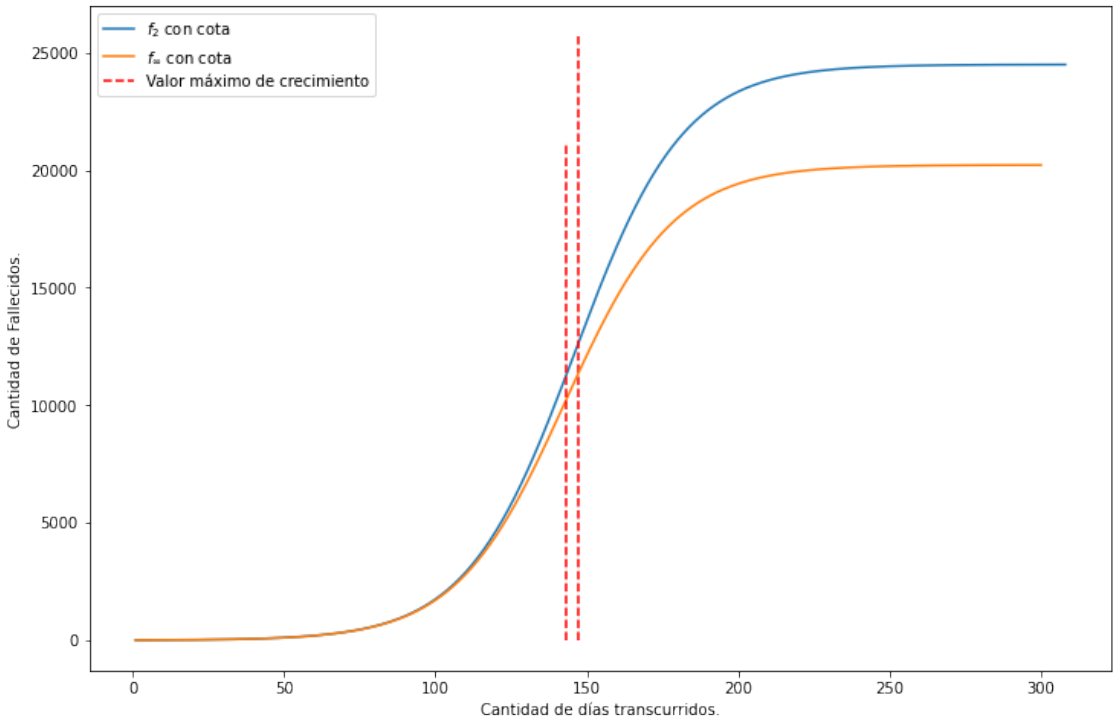
[15]:
```
plt.figure(figsize = (12,8))
# plt.plot(Error._Dia,Error._Fallecidos,label='Curva de fallecidos.')

Error.__modelo__(585680.44 , 76952.25, 18.29, obj='Fallecidos')
Error.__get_n_max__()
D2 = np.arange(1,int(Error.n_max*2.1)+1)
N = Error.n_max*np.ones(len(D2))
L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
plt.plot(D2,Error.modelo(D2), label = r'$f_2$ sin cota')
plt.plot(N,L2, 'r--')
# plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_2$ sin cota')
Error.__modelo__(84505.39, 11098.93, 18.34, obj='Fallecidos')
Error.__get_n_max__()
D2 = np.arange(1,int(Error.n_max*2.1)+1)
N = Error.n_max*np.ones(len(D2))
L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
plt.plot(D2,Error.modelo(D2), label =  r'$f_\infty$ sin cota')
plt.plot(N,L2, 'r--')
# plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_\infty$ sin cota')
Error.__modelo__(24500, 3410.44, 17.95, obj='Fallecidos')
Error.__get_n_max__()
D2 = np.arange(1,int(Error.n_max*2.1)+1)
N = Error.n_max*np.ones(len(D2))
```

```
L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
plt.plot(D2,Error.modelo(D2), label = r'$f_2$ con cota')
plt.plot(N,L2, 'r--')
# plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_2$ con cota')
Error.__modelo__(20227.32, 2840.78, 17.95, obj='Fallecidos')
Error.__get_n_max__()
D2 = np.arange(1,int(Error.n_max*2.1)+1)
N = Error.n_max*np.ones(len(D2))
L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
plt.plot(D2,Error.modelo(D2), label = r'$f_\infty$ con cota')
plt.plot(N,L2, 'r--', label = 'Valor máximo de crecimiento')
# plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_\infty$ con cota')


plt.xlabel('Cantidad de días transcurridos.')
plt.ylabel('Cantidad de Fallecidos.')
plt.legend()


# Error.__get_n_max__()
# D2 = np.arange(1,int(Error.n_max*2.1)+1)
# N = Error.n_max*np.ones(len(D2))
# L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
# plt.plot(D2,Error.modelo(D2), label = 'Curva de ')
# plt.plot(N,L2, label = 'Valor máximo de crecimiento')


# plt.title('Curva de acumulada de {}'.format(data_label))
# plt.legend()
plt.show()
```



[16]:
```
plt.figure(figsize = (12,8))
# plt.plot(Error._Dia,Error._Fallecidos,label='Curva de fallecidos.')

# Error.__modelo__(585680.44 , 76952.25, 18.29, obj='Fallecidos')
# Error.__get_n_max__()
# D2 = np.arange(1,int(Error.n_max*2.1)+1)
# N = Error.n_max*np.ones(len(D2))
# L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
# plt.plot(D2,Error.modelo(D2), label = r'$f_2$ sin cota')
# plt.plot(N,L2, 'r--')
# # plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_2$ sin cota')
# Error.__modelo__(84505.39, 11098.93, 18.34, obj='Fallecidos')
# Error.__get_n_max__()
# D2 = np.arange(1,int(Error.n_max*2.1)+1)
# N = Error.n_max*np.ones(len(D2))
# L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
# plt.plot(D2,Error.modelo(D2), label =  r'$f_\infty$ sin cota')
# plt.plot(N,L2, 'r--')
# plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_\infty$ sin cota')
Error.__modelo__(24500, 3410.44, 17.95, obj='Fallecidos')
Error.__get_n_max__()
D2 = np.arange(1,int(Error.n_max*2.1)+1)
N = Error.n_max*np.ones(len(D2))
L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
```

```
plt.plot(D2,Error.modelo(D2), label = r'$f_2$ con cota')
plt.plot(N,L2, 'r--')
# plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_2$ con cota')
Error.__modelo__(20227.32, 2840.78, 17.95, obj='Fallecidos')
Error.__get_n_max__()
D2 = np.arange(1,int(Error.n_max*2.1)+1)
N = Error.n_max*np.ones(len(D2))
L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
plt.plot(D2,Error.modelo(D2), label = r'$f_\infty$ con cota')
plt.plot(N,L2, 'r--', label = 'Valor máximo de crecimiento')
# plt.plot(Error._Dia,Error.modelo(Error._Dia),label=r'$f_\infty$ con cota')


plt.xlabel('Cantidad de días transcurridos.')
plt.ylabel('Cantidad de Fallecidos.')
plt.legend()


# Error.__get_n_max__()
# D2 = np.arange(1,int(Error.n_max*2.1)+1)
# N = Error.n_max*np.ones(len(D2))
# L2 = np.linspace(0,np.max(Error.modelo(D2))*1.05,len(D2),dtype=int)
# plt.plot(D2,Error.modelo(D2), label = 'Curva de ')
# plt.plot(N,L2, label = 'Valor máximo de crecimiento')


# plt.title('Curva de acumulada de {}'.format(data_label))
# plt.legend()
plt.show()
```


f_2 con cota
f_∞ con cota
Valor máximo de crecimiento
25000
20000
15000
10000
5000
0
Cantidad de Fallecidos.
0
50
100
150
200
250
300
Cantidad de días transcurridos.